A New Land Battle for the Theater War Exercise

THESIS

Marlin Allen Ness
Captain, USA

AFIT/GE/ENG/90J-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY
Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/90J-01

A New Land Battle for the Theater War Exercise

THESIS

Marlin Allen Ness
Captain, USA

AFIT/GE/ENG/90J-01

AFIT/GE/ENG/90J-01

A New Land Battle for the Theater War Exercise

THESIS

Presented to the Faculty of the
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Marlin Allen Ness, B.S.
Captain, USA

June 1990

Preface

The goal of this thesis was to develop a theater level land combat model from scratch. Initial research provided information about the latest land combat modeling theory, army operating procedures, tactical and strategic operations of units in the field, and initial constraints and assumptions. These formed a sound basis for the requirements and specifications that would model sufficient and necessary land combat processes. Detailed design and implementation of the specifications using object oriented design methods occured in five phases.

This thesis provides essential background material, the development of requirements and specifications, and the initial and detailed design of a theater level land combat model. The final solution is a very powerful tool that is flexible, powerful, easy to understand, and credible. It provides a solid land combat modeling foundation that can be easily enhanced or modified.

I am indebted to my thesis advisor, Major Mark A. Roth, for his assistance and guidance in the development of this thesis. I also wish to thank Major Michael Garrambone and Lieutenant Colonel Skip Valusek for their thorough coverage of the written manuscript. Finally, I must thank my wife Helga, who never ceased to provide support and understanding, and my sister Christa who watched Marlin Jr.

Marlin Allen Ness

Table of Contents

List of Figures

List of Tables

Abstract

The Theater War Exercise (TWX) is a four day, theater level, two sided, airpower employment decision-making exercise run at the Air Force Wargaming Center, Maxwell AFB, Alabama. Decisions made by the exercise participants are fed into TWX's air and land battle simulation programs, which then simulate the employment of the air power strategy, doctrine, and war fighting principles inherent in those decisions. TWX was limited for use as a potential joint exercise because of the existing land battle simulation. The old land battle was a standalone simulation whose only interface to the air battle was the passing of sortie information through an intermediate database relation. Additionally, a carefully prepared programming script was used to ensure that the land units performed reasonably. This made it impossible for the land game to react to the air play in any way except through a slowing of land units when on their preprogrammed paths.

The goal of this thesis effort was a complete development of a new land combat model program. This was accomplished through a determination of the proper levels of player participation and the required level of aggregation for the land units. A six step object oriented design process was used along with application prototyping for program development.

An aggregated interactive theater-level land combat model and its implementation in Ada is described. The model simulates doctrinal planning and decision making operations conducted at Army Group level. It provides credible land combat processes, unit movement and attrition, logistics, and intelligence based on player inputs, air–land unit interactions and terrain characteristics.

Overall, the program is an easily modified, easily configured simulation that is potentially useful at any level of combat. It is a very powerful tool that is flexible and credible. It provides a solid land combat modelling foundation that contains all of the key processes found in any other model of its size. Because of the object oriented development methodology, enhancements require minimum effort on the part of the developer and can be easily integrated into the program.

A New Land Battle
for the
Theater War Exercise

I. Introduction

1.1 Background

The Theater War Exercise (TWX) is a four day, theater level, two sided, airpower employment decision-making exercise run at the Air Force Wargaming Center, Maxwell AFB, Alabama. It is based on a combat scenario conducted in NATO's Central Region between Red and Blue Players played by senior Air Force officers. The Red players representing Warsaw Pact forces are played by Air University faculty and staff and the Blue players representing NATO forces, are played by seminar groups consisting of 8-15 players. The game controller provides operational guidance, instructions and operates the computer simulation program of the same name, TWX(13). The original TWX simulation program was written in 1977 at the request of the USAF Chief of Staff. The decisions made by the exercise participants are fed into both TWX's air and land battle simulation programs, which then simulate the employment of the air power strategy, doctrine, and war fighting principles inherent in those decisions(13, 25).

Theater War Exercise participants play out the roles of the command and staff at the Allied Air Forces Central Europe (AAFCE), Allied Tactical Air Force (ATAF), Northern and Central Army Group (NORTHAG and CENTAG) and at the Commander-in-Chief Allied Forces Central Europe (AFCENT) levels of play(25, 49). Individual players conduct the planning and issue directives based on their assessment of the situation provided by game controllers. AAFCE players are primarily concerned with logistics, relocation and the reroling (reassignment) of theater air forces, air directive publication and staff support. An overall strategy is devised for implementation by the ATAF commanders and staffs. Figure 1 shows the command and control relationships between the Air Force and

Army units(50:47) while Figure 2 shows the various roles for AAFCE players during the exercise(13:4).

ATAF level players are primarily concerned with allocating resources based on AAFCE guidance(25). Although targeting and packaging of aircraft actually occurs at the Air Tactical Operations Center (ATOC) and Air Support Operations Centers (ASOC), the detailing of aircraft is conducted by the players at the ATAF level in TWX(25). This was done strictly for game purposes, as the ASOC and ATOC operations are not explicitly modeled in the air or land portions of TWX. Figure 3 shows the player roles for the ATAFs(13:5).

Inputs for the simulation are entered through screen based graphics and are provided to the databases through a relational database management system. After the input is finished, the air war program is run and results are again fed to the database. Next, the land battle simulation program is run, with the results being passed to the database. Each computer run accounts for twelve hours of combat. In general, two runs are made daily and simulation results are given to the players each day based on their inputs. These results are then used to prepare the next day's battle plans, with critiques conducted at the conclusion of the exercise. Because the data and the simulation are kept unclassified and the algorithms of the simulation have never been verified or validated, no direct correlation between any given simulation run and reality can be made. However, the results are deemed credible by the participants. The exercise does provide players a feel for airpower strategy, doctrine and required decision making that must be made at AAFCE and the ATAFs during war(13).

1.2 Problem

Although, the air portion of the Theater War Exercise was fairly mature, TWX was limited for use as a potential "joint" exercise because of the existing land battle simulation component. The old land battle was a standalone simulation whose only interface to the air battle was the passing of sortie information through an intermediate database relation. A carefully prepared programming script was used to ensure that the land units performed in reasonable manner, but this made it impossible for the land game to react to the air play in any way except through a slowing of land units when on their preprogrammed paths(57).

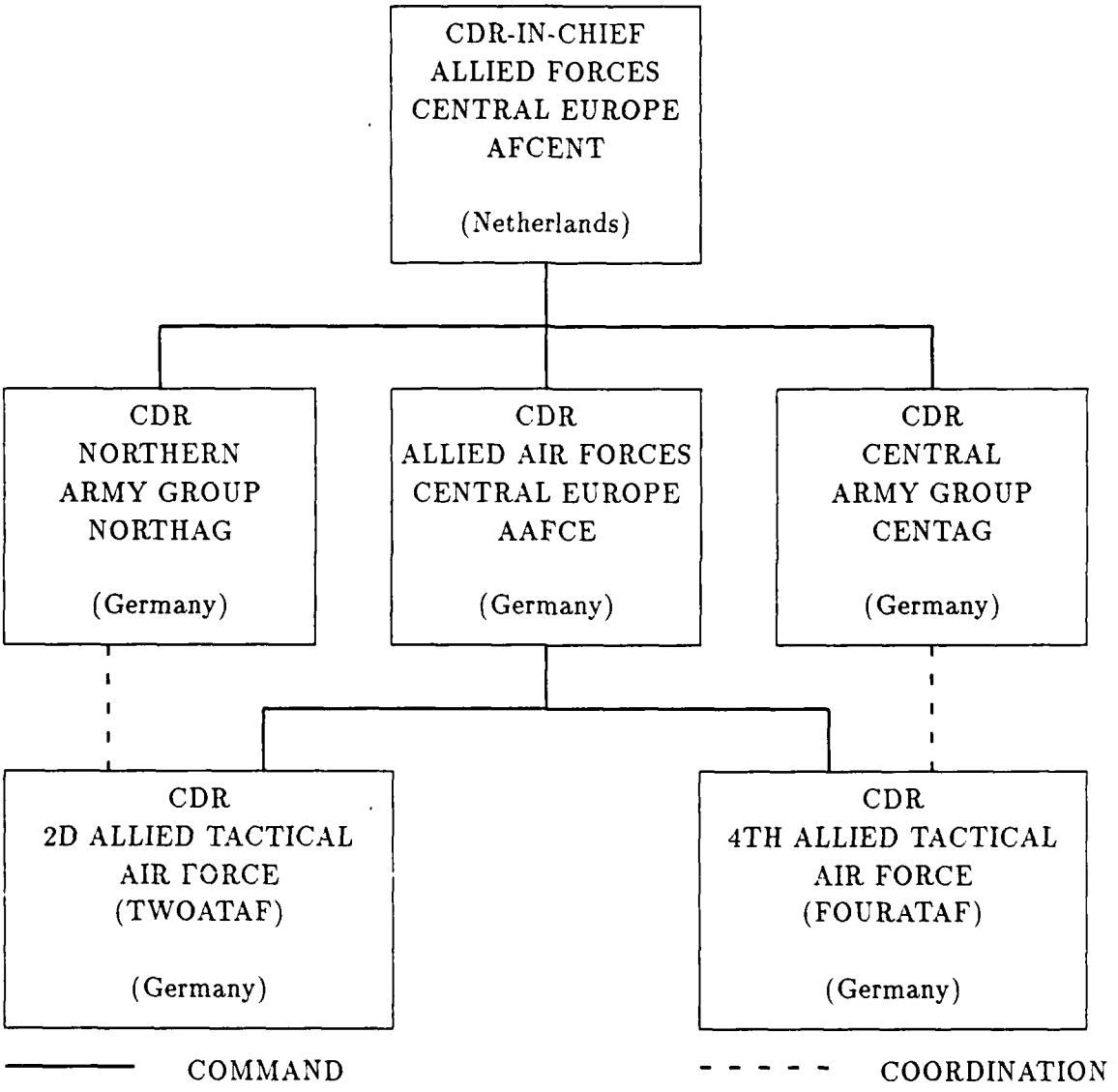

Figure 1. NATO Command and Control.

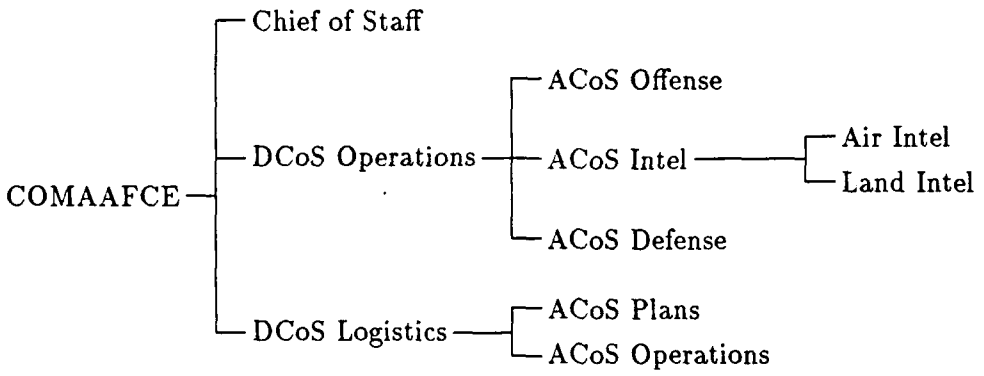

Figure 2. AAFCE Player Roles.

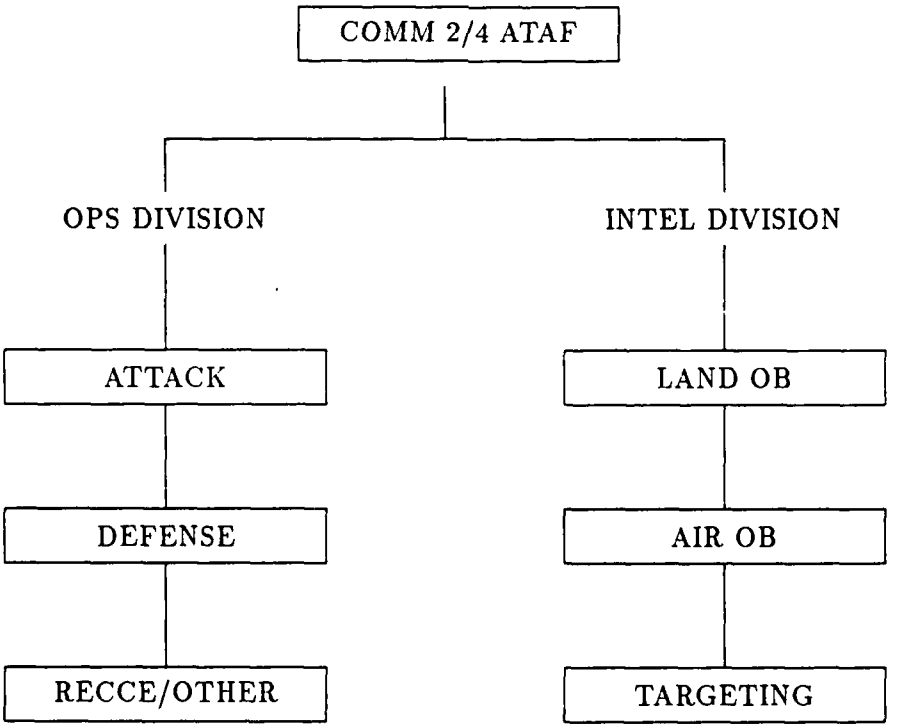

Figure 3. ATAF Player Roles.

This was neither satisfactory for a simulation nor for an educational exercise. These problems were considered the major weaknesses of the Theater War Exercise. Additionally, players were frustrated with their inability to affect the land battle and with the unreality of the land simulation at their level of play. This thesis effort attempted to correct these problems by replacing the land portion of TWX with a credible ground combat model.

1.3 Study Constraints and Assumptions

Development and acceptance of the new land battle by the Air Force Wargaming Center was based on the following assumptions, that:

- The land battle would be considered adequate if land units were controllable, if unit actions were doctrinally sound and credible, and if the battle output results were considered credible.
- The air and land units interfaced through standard operational practices.
- No modifications of the internal operations of the air war were required and that connectivity between the new land battle and the air war would be built in as needed. Additional structures were provided for future enhancements.
- Command, control and communications would be modeled by the player interaction in the game and not by the computer simulation.
- Verification and validation of the new land battle would be conducted by the Air Force Wargaming Center.

The remainder of this section discusses army units, the three levels of war, and joint Army and Air Force operations. This effort determined program constraints and assumptions for the player levels and the lowest level of aggregation for the units.

1.3.1 Army Units. The highest levels of military structure for United States military units are the five unified commands. The five unified levels of command, also called theaters of war, are often broken down into theaters of operations when necessary. Army groups may be found between the theaters of operations and armies or corps(67, 49). Army

groups are established at the direction of the theater of operations commander with the approval of and in coordination with the theater of war commander(67). Army group operations are concerned with the "deployment, maneuver and fires of land combat power over extended terrain and the integration of all Army and other service support into the operations"(67:5-2). "These army groups are organized to plan and direct the major operations of the campaign and thus have no sustainment function or assets"(67:5-1). The actual implementation of operations based on the army group commander's guidance is left to the subordinate commanders(67). There are two Army Groups in NATO each composed of several corps: the Northern Army Group (NORTHAG) and Central Army Group (CENTAG). Figure 4 shows an Army Group Headquarters(67:5-4). In this theater, an army group controls from two to five corps(67).

Corps are the largest tactical units for which there is complete Army doctrine and "for which purely tactical responsibilities are still conceivable"(67:vi). They are the basic units which are used by echelons above corps (EAC)to conduct maneuver at the operational level of war. Corps are designed for conducting sustained operations for considerable periods of time and are often tailored for specific missions(29). They typically have between two and five divisions, many combat service and combat service support brigades and other non-divisional units. In Europe, corps normally have two divisions, many subordinate support brigades, and an armored cavalry regiment. The corps artillery has at least two brigades(28).

The division is the largest United States organization that actually trains together as a combined arms team. Divisions usually fight as part of larger forces(27, 29). Brigades are the major combat units in a division. They usually have at least two battalions, while divisions have eight to eleven maneuver battalions and three to four field artillery battalions(29). A brigade's chief tactical responsibility is synchronizing the plans of actions of subordinate units to accomplish any desired task of its higher headquarters(29). They use all means available to attack, destroy, disrupt and stop enemy forces(49).

Battalions perform a single tactical mission as part of an overall brigade plan(27). Individual initiative is allowed as long as the battalion plan remains within the context of the overall brigade concept and mission orders. Battalions have the specific missions of

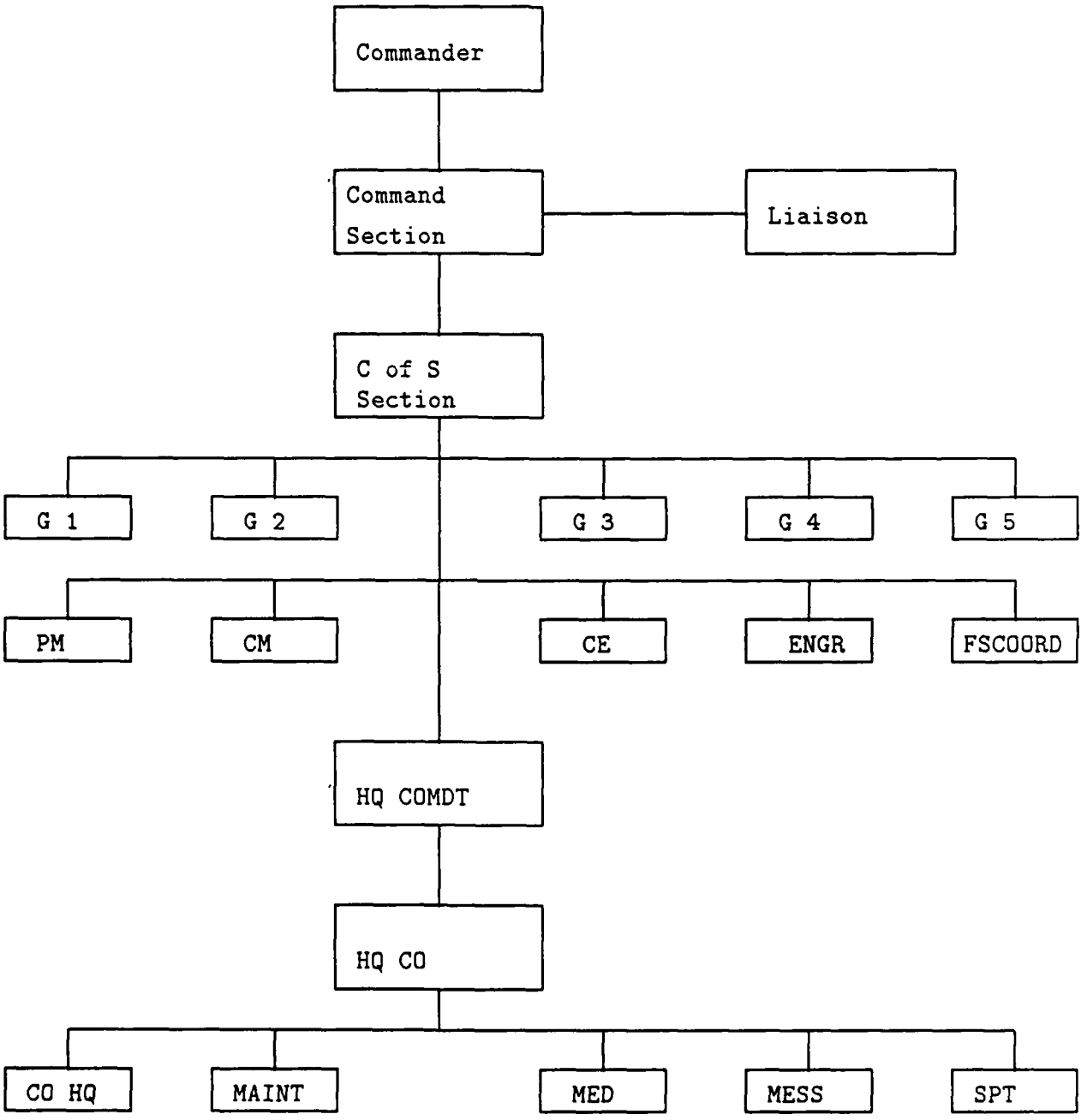

Figure 4. Army Group Headquarters.

Table 1. Areas of Influence.

Level of Command	Time	Approximate Distance Beyond FLOT
Battalion	0-3 hrs	5 km
Brigade	0-12 hrs	15 km
Division	0-24 hrs	70 km
Corps	0-72 hrs	150 km
Echelons Above Corps (EAC)	72+ hrs	150+ km

attacking, defending, exploiting, delaying, pursuing, counterattacking and moving to new missions. Battalions or cavalry squadrons have three to five companies, troops or batteries each(29, 49). Companies, troops, or batteries are the classic elements of all battalions and perform tactical missions as directed by their next higher headquarters(27, 29).

The actual conduct of the battles in a unit's purview occurs in the unit's assigned area of operations. The unit's area of interest extends as far as any enemy units or activities which might affect its own activities and operations(29). Areas of influence (where a commander can influence the enemy and acquire targets) extend into enemy territory approximately 70 kilometers for a division, to 150 kilometers for a corps, and past 150 kilometers for echelons above corps. Divisions can act as independent units, but they normally fight as part of a corps. Divisions typically allocate resources for combat for up to 24 hours and plan for combat for 72 hours. Corps typically allocate resources for up to 72 hours. General guidelines for areas of influence beyond the Forward Line of Troops (FLOT) are in Table 1(64).

The old Theater War Exercise did not represent these military structures or command relationships correctly.. Although units were used in a mock simulation and were subordinate to other units, specific command relations were not defined and areas of influence were not established. This thesis effort corrected these flaws. The conduct of operations for units and the development of assumptions and constraints were based on the three levels of warfare and the three types of tactical operations.

1.3.2 Levels of Warfare. The basic concept of fighting for the United States Army is the AirLand Battle doctrine. The AirLand Battle consists of three basic levels: strategic, operational and tactical.

> National strategy (strategic) is the plan or expression of the coordinated use of national power which includes political, economic, psychological, and military power plus national will during peace, crisis, or war to secure national objectives(67:1-1).

Strategic level of war objectives are accomplished by the unified commanders through the conduct of operations. The Strategic level of War is not played in the Theater War Exercise.

"Operational art involves the design and execution of campaigns and major operations; tactics, the planning and conduct of battles and engagements"(67:2-1). The operational level of war is concerned with gaining an advantage over the enemy through the maneuvering of large forces in some area of operation. *It is the employment of forces* through major operations to obtain an objective based on the campaign plan(50). The campaign plan specifies how the commander of a theater of operations uses all available resources to conduct war. Operational planning occurs mostly at joint, theater and corps level and focuses on the actual missions and operations that are to be conducted by subordinate units(27, 29).

The tactical level of war is concerned with the actual destruction of the enemy by application of force through execution of the plan. This is usually done below corps level. The theater commander selects the overall plan of action for a conflict, which is then transmitted to subordinate units for execution(29). Tactical plans pertain to shorter term goals than operational plans, but like operational plans, consist of close, deep and rear operations.

A commander and his staff at any level is always aware of the status, capabilities and missions of all units down to at least two levels below his own, irrespective of the level of warfare. Corps commanders know the status of their subordinate brigades and division commanders track their subordinate battalions. Commanders often directly control the

operations and capabilities of their next lower level units to ensure success of the overall plan(29:38). The land commander's success in combat is enhanced by joint operations. This is particularly true for joint Army and Air Force operations. Joint service operations are also based on the strategic and operational plans of the theater commander(49).

1.3.3 Close, Deep and Rear Operations. "Corps and divisions conduct mutually supporting operations simultaneously in three areas–close, deep and rear"(29:33).

Close operations focuses on the opposing forces in combat at the FLOT and includes any "readily available reserves of both combatants"(29:36). All means available are used by the combatants with a concentration on maneuver, fire support, assignment of a main thrust to a subordinate unit, ready use of reserves, and the use of obstacles and planned supporting fires. The general intent of the battle is to maneuver for advantage and to use all assets to attrit the enemies rear and flanks(29).

"The commander supports his basic scheme of maneuver with deep operations against specific enemy forces in depth that threaten his success"(29:37). Key elements of deep operations include: diversion, disruption, delay, and destruction of the second echelon forces by limiting the enemy's freedom of action, deceiving the enemy, isolating, immobilizing and weakening defenders in depth, and attacking critical targets with organic and supporting aerial, artillery and missile weapons(66). In the defense, deep operations are used to deter enemy combat buildup(29).

Rear operations keep the commander's freedom of action open and help ensure continued support for the front line battles by protecting rear areas(29, 66). Key elements of rear operations include rear area security and the free flow of logistics through the use of reserves, military police, fire support, engineer units and alternate routes.

1.3.4 Joint Army and Air Force Operations. In NATO, the Allied Air Forces Central Europe (AAFCE) is subordinate to the Allied Forces Central Europe (AFCENT). Command structures of the Northern and Central Army Groups (NORTHAG and CENTAG) are at a comparable level and work directly with ATAFs.

> The army group commander, in coordination with corps commanders, identifies his needs for air assets (reconnaissance, close air support, interdiction and air lift). He also participates in the targeting process by nominating targets for engagement. These actions are done to influence the air apportionment and allocation process. In sum, the army group commander must involve himself in the planning process for the apportionment, allocation, and targeting for air assets to ensure that the available air assets support his concept of operations(67:5-8).

Air interdiction is the attack of the enemy's combat forces that would have a near term effect on friendly forces before those forces can be used(66). Battlefield air interdiction (BAI) is applied "against land force targets that could have a near term effect, but are not in close proximity to friendly forces"(66). BAI is conducted in synchronization with land combat maneuver forces and probably extend out to about 100 kilometers beyond the FLOT(49). BAI is usually a percentage of the total AI mission and is composed of a separate list of targets(66). The percentage of BAI and close air support (CAS) are normally provided to a corps based on the corps' requests and the prioritized target listing at echelons above corps(EAC)(16:102). Air Force support is consolidated and prioritized by the Tactical Air Control Center (TACC) in coordination with the Battlefield Coordination Element(BCE) co-located with the TACC(66). General guidance for priorities to the BCE comes from the land component commander. Although BAI and CAS targets are prioritized by corps staff based on inputs from subordinate units, the army group commanders are the real BAI asset managers(49).

ASOCs are found at Corps Tactical Operations Centers (TOCs). Figure 5 displays the general command, control and coordination lines between Army and Air Force units in AFCENT(68:17A). ASOCs and Tactical Air Control Parties (TACP) assist corps in identifying and prioritizing appropriate targets. ASOCs coordinate and direct CAS and tactical air reconnaissance (TAR) support for corps(66). There is a continuous process of updating and prioritizing targets as new information is obtained.

Because present joint Army and Air Force operations manipulate large forces at many echelons, this was also necessary for the simulation. All important staff functions played in the Air portion of the Theater War Exercise pertain to the echelon above corps level of play. Actions that are pertinent to the simulation should be modeled at this level. Because

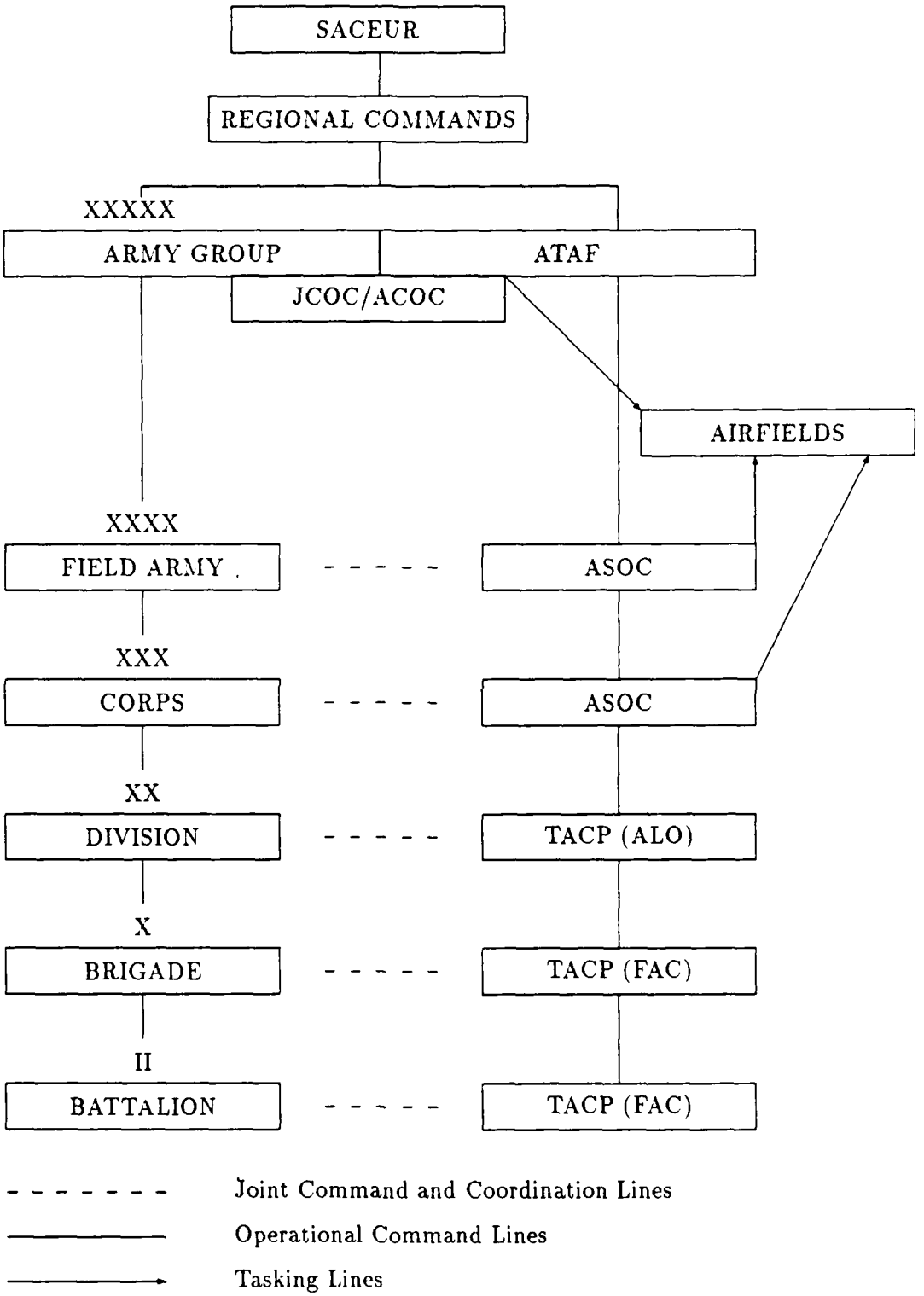

Figure 5. NATO Command and Control Agencies.

TWX was developed to "teach operators and planners about the threat and application of force," it was important that players not become too involved at too low a level of play(25:4). This philosophy was also applied to the new land battle. As mentioned earlier, the strategic level of war is not represented at all in this model. High level operational decisions are made by the actual players interactively in the game rather than by the computer simulation. A basic scenario played at the tactical level of war is portrayed by the computer simulation derived from player inputs. Player inputs are based on proper planning, coordination, and decision-making by the commanders and staffs(24).

The lowest level Air Force operations and planning for the air war simulation occurred at the Allied Tactical Air Forces (ATAF) level of play. This corresponded most closely to the army group level, which is directly supported by an ATAF. Based on this information, a major assumption of this thesis effort was that all planning and inputs for the land battle would only be from the perspective of an army group commander or staff controlling their subordinate units. This constraint adheres to the original precepts and objectives of the Theater War Exercise and follows logically from present operations procedures w army groups are part of the command structure. Figure 1 showed a summary of the levels where players act as the commander or staff at the AAFCE, ATAFs and Army Groups. The next section discusses aggregation based on this constraint.

1.3.5 Land Combat Level of Play. "Simulation is defined as the development of a mathematical-logical model of a system and the experimental manipulation of the model on a digital computer"(5:7). This implies a computer simulating land and air combat. To develop a workable land combat model, it was important that aggregation take place. Aggregation is the process of representing a large amount of detail with less detail. It is a fundamental concept of modeling at low resolution.

One of the basic assumptions in developing the land combat model was that everything on the battlefield could not be modeled. Simplicity was important in model development and less important detail had to be left out. Common important features of a good model include: aggregating the forces in the model, scaling the power of the forces relative to one another, weather and terrain conditions, tactical postures, air and

land unit interaction, and logistics(44). No model can mimic reality exactly, especially a combat model. Some concepts may be extremely difficult to model: command, control and communications; intelligence; and/or logistics. Unnecessary details are often left out when not required. As mentioned previously, command, control and communications are modeled by player interaction prior to the simulation run. Although connectivity was provided in the model for future enhancements in logistics, intelligence, etc, the lowest level of aggregation had to be determined. Requirements specifications (exactly what is modeled in the new land battle program) are discussed in greater detail in Chapters III and IV.

Aggregation of the lowest level details was used in the old Theater War Exercise and had to be applied in the new land combat model. Some detail had to be omitted or the model would have become too unwieldy with possibly millions of entities (combat units). Even a hundred entities was too large a number for an army group commander or staff to be concerned with. It had to be ensured that the players concentrated on the operational rather than the tactical level of play.

In this theater there are at least two army groups, each with two to five subordinate corps. Each corps has at least two to four divisions. Each division has at least four maneuver brigades. This gave a minimum of 160 units. It was feasible that each unit could have up to 15 attributes including unit designation, type of unit, mission, mission location, firepower, supporting units, intelligence index, breakpoint, combat power, higher unit, and lower units. These considerations and the constraint that the game be played from the perspective of an army group staff provided the best indication of the proper level of aggregation. In order to keep the model simple and considering the time required to actually run a model, it was most reasonable to play only with division level and corps non-divisional units. This was consistent with the old Theater War Exercise and it is the normal mode of operations for an army theater commander and staff, who would always be aware of the complete status of all units down to division and corps support brigades in his theater. Commanders and staffs at the army group (AG) level are required to plan operations for approximately 40 major units. This directly corresponds with the operational level of play expected of an AG level staff which "directs assigned or attached units to destroy the enemy and to gain or retain control of terrain" as well as other missions(67:5-2). Actual

implementation of the directives is left under the control of the simulation. The lowest unit of aggregation for the new land model are divisions and major corps non-divisional units (brigades and regiments). All attributes below the division are aggregated into the attributes of the division entity or as a corps support brigade.

1.4 Approach and Methodology

As mentioned previously, everything on the battlefield can not be modeled. This thesis effort attempted to design and develop an operationally credible land combat model for the Theater War Exercise with command resolution at the Army Group level. A major assumption was that the major corps non-divisional units (e.g. cavalry, field artillery, air defense, aviation and engineer) and divisions are the lowest level of aggregated units necessary.

The overall strategy for solving this thesis started with requirements specification and an object oriented design focusing on the planning and operations aspects faced by an Army Group commander or staff involved in conflict in Central Europe. The primary focus of this effort was in simulating conflict based on the inputs of commanders and staffs at army group level. The following objectives supporting this strategy were solved and are presented in Chapters III through V:

1. Develop necessary unit attributes and interfaces for the land combat simulation. A requirements analysis of necessary unit attributes, capabilities, planning and operations aspects of combat units was conducted. Necessary unit information, capabilities and operations planning attributes were taken from the latest theory on land combat modeling, simulation of systems theory and US Army Field Manuals and Operating Procedures. This is discussed in Chapter II. The program was derived from this information and the following player's objectives(24:3):

   (a) Analysis of the area of operations and mission.

   (b) Future operations planning.

   (c) Requesting targeting priorities from ASOCs and AAFCE.

(d) Reacting to enemy operations.

(e) Issuance of orders to subordinate divisions. Orders include mission, force composition, etc, based on intelligence received and commander's guidance.

These player's objectives were combined with the requirements analysis to determine exactly what was needed for the user's inputs.

2. Determine the objects and operations for a land battle program. This is presented in Chapter IV using object oriented design methods.

3. Determine the best language for the simulation. Analysis showed that the high level simulation languages Simulation Language for Alternate Modeling (SLAM) II, Ada, C and Fortran were feasible languages for the model development. Ada was used for program design and initial development. Very strong reasons for the use of Ada as the development language can be found in Appendix B. Ada was used with the concurrence of the Air Force Wargaming Center.

4. Development of a land combat simulation that could:

   (a) Move units on the battlefield based on player directives and game attributes.

   (b) Perform attrition between enemy land and air units.

   (c) Provide combat support and combat service support operations and functions between units.

   Code development for these objectives was based on the latest doctrine, combat modeling theory and the requirements of the Air Force Wargaming Center. The code is author original, developed using object oriented design methods. Rationale for code development is discussed in Chapters III, IV and V.

Application prototyping was the software development methodology used for this thesis. Basic reasons for the use of this methodology can be found in (70:9). Pertinent reasons for using application prototyping include:

- No requirements specifications. Only a general idea of what was required was available.

- The user interface for the land combat portion was highly interactive.
- Modifications were required once the capabilities of the system were known.
- Minimum development time was available.

Intermediate results of this thesis were presented quarterly to the Air Force Wargaming Center for review, discussion, and changes. The power of Ada, application prototyping and the 4GL tools of a relational database system helped to ensure the success of this thesis effort. Chapter III provides discussion on additional enhancements to the original thesis proposal.

1.5 Materials and Equipment

The land combat simulation was developed using the existing air portion of the TWX simulation. A relational database management system was used for database management and for developing the screen based user interfaces. Two Zenith Z-158 microcomputers, a Zenith Z-386 microcomputers, and a Sun 386i Work Station and software were provided by the Air Force Wargaming Center for use. Only the 386 microcomputers were used for development. All other required hardware and software was provided by the Air Force Institute of Technology. A Janus Ada compiler (version 2.1.2) was used for initial compilation. This had to be replaced with a Verdix compiler for the Sun and then a 386 Janus compiler for the Zenith.

1.6 Sequence of Presentation

This thesis presentation is organized into five additional chapters based on the chronological development of the final program. Chapter II provides general insight on land combat modeling. Based on the most important concepts of the theory of land combat modeling, Chapter III discusses the specifications and requirements for the new land battle simulation. Chapter IV is the initial design of the program and Chapter V describes an object oriented detailed design and implementation of the land battle program. Chapter VI contains a summary, conclusions and recommendations for future work. Several

appendices provide additional information about databases, Ada and problems with the MS-DOS operating system.

II. Land Combat Modeling

2.1 Introduction

The purpose of this chapter is to describe the literature providing pertinent discussion and definitions of land combat modeling as it applies to war games. It focuses on the key processes of combat and how these processes are typically modeled. The next several subsections provide a general discussion of some of the difficulties and general characteristics of models. This discussion is followed by a presentation of land combat specific topics. The chapter focuses on the most critical aspects of combat that could be used in the design of an interactive Land Battle (LB) program. Chapter III then provides the requirements specifications for the LB program based on concepts presented in this chapter.

2.2 Modeling in General

Although modeling covers a very large area, the primary focus here is on war games. The term war game often applies to analytic games, computer games, interactive computer games, computer assisted war games, manual war games and military exercises(40). Of particular interest here are interactive computer war games which provide experience and proficiency just short of actual combat(21). A war game is different than other simulations in that the primary purpose of a war game is often to provide insight, experience, and operational expertise for players or participants. Players acting as commanders and primary staff are generally encouraged to make intelligent "exercise" command and control decisions and to learn from the results of those decisions. This is often done interactively ("man-in-the-loop") with a computer program which simulates combat based on those player decisions. The Theater War Exercise is one such "man-in-the-loop" war game. Players provide command and control decision inputs prior to every simulation run. The simulated combat environment is provided as an opportunity for players to gain experience in making command and control decisions in a nonlethal environment. Credible play was a primary goal in its development. The important concept here is that "for the purposes intended it (the model) needs to be like the real thing"(21:35).

Unfortunately, modeling combat is not easy and there is often no single correct answer. The plethora of documents, articles, books and academic research (see bibliography) which have discussed the basic difficulties of land combat modeling have not produced a fundamental theory nor a universally accepted verified and validated land combat model. It is simply not possible to model all the interactions that might occur. Of particular difficulty is modeling the incredibly complex human decision making processes that occur at every level, every minute on a battlefield. Applicable combat data of all types is frequently ambiguous, incomplete or incorrect. The processes are not fully understood, the inputs are often not correct, and interpretation of the results leaves widespread disagreements. Furthermore, it is often difficult to determine exactly what might be classified as most important or extraneous in the model, in the data and in the simulation(47). The higher the level of consideration, the more difficult it becomes. "It follows that there is no such thing as an all-purpose theater-level model"(40:VI-1). At best, a very general approach is taken to depict combat.

The next few sections focus on general features and aspects of simulation and modeling. Further information about land combat modeling can be found in (4, 63, 34, 35, 61).

2.2.1 High Resolution and Aggregated. There are two general categories of combat modeling: high resolution and low resolution or aggregated. High resolution models examine individual entities on the battlefield focusing on individual soldiers, weapons systems and their effects in one-on-one engagements. High resolution models usually focus on modeling brigade and lower unit interactions, particularly from battalion to platoon level. Although high resolution models focus on the lower level details, it should be noted that more detail is not always better(61).

Aggregated models group the many attributes of these entities into a parameter to represent this collection. "As we try to model larger forces at division level or higher, the sheer number of combatants and weapons systems makes it impossible to maintain individual item resolution"(34:1-3). Of course, all the entities and attributes of those entities below the level of aggregation are lost or given a sub-unit status. This is not a

problem as long as the focus for the model is not on the low level entities. Theater level and higher models are typically aggregated models.

2.2.2 Deterministic and Stochastic. Both high resolution and aggregate models can be deterministic or stochastic. A deterministic model does not have random number generation or variation and will always produce the same results for the same given set of inputs. A stochastic model is one in which a random number or random variation is used to achieve probabalistic results for every run. It is never expected that any two runs would produce exactly the same result. Rather, statistics are used to gather general information from the model. Thus, many simulation runs are required to get an "average" for the samples. This is done for analysis purposes when conclusions must be drawn from the simulation results. War games with actual player participation are stochastic, due to the randomness of the players, although the actual simulation for a given set of inputs might be deterministic. Whether a war game is stochastic or deterministic is not important, so long as the insights to be gained by the players can be gained through the results of the simulation.

2.2.3 Discrete Event Simulation. The actual control of the simulation events and processes are either discrete, continuous or a combination of the two. Although the actual system is seldom completely continuous or discrete, it is usually the case that the system is modeled predominantly as either one or the other. Discrete event simulation is the most common type and will be the only one presented here.

Discrete event simulations focus on modeling the events in the system using discrete events in time. Changes are generally discontinuous in nature. The point in time at which a change may take place in the system is usually called an "event" or "event time". This is when the variable changes occur. Some type of logic is associated with each event and the system is modeled by the changes in state of the system(54). Changes to attributes of entities may also be caused by the events(33). Process interaction and event-scheduling are the two major approaches used. Process interaction focuses on processes which are time-ordered events, activities and delays that have some relation to an entity. These

processes are usually quite complex, the process method is not often used, and will not be discussed further(45).

Event-scheduling focuses on the effects on system state caused by events. Changes in events or event occurrences making up the model cause state changes. Entities represent the objects within the boundaries of the system. The fixed time step method is one of the most common methods for program control using event scheduling. Events or groups of events occur at fixed time intervals in the model. Fixed time step models are considered discrete models because the state variables change at discrete events in time(35).

Fixed time steps allow battle periods to range from minutes to days. Typically a cycle will be twelve or twenty-four hours for a theater level model. Cycles may be reduced in duration to more closely model the combat process or to gain better control of the events in the simulation. Some models update corps once per day and theater levels once every four days(61). Division level units in combat should not make vast maneuvers or complete missions in time steps under twelve hours. It is quite common and particularly appropriate to model large scale models using time step simulation processes(61).

2.2.4 Database Management. Once a simulation is operational, there must be some means for the simulation to acquire, manipulate and output data. Although some simulations may have very little interaction with data, or only have simple data handling requirements, simulations involving combat often require large quantities of data, vast computer resources and extensive data handling. Data use and manipulation without a database management system is often quite difficult. See Appendix A for a discussion of database management systems, their advantages, and why one is used with the land battle program.

2.2.5 Languages for Design and Development. Although simulations are often developed using a specialized simulation language like GPSS, SIMSCRIPT II.5 or SLAM II, it is often the case that for practical purposes

> the simulationist must use a high-level programming language (HLPL) such as C, Pascal, and FORTRAN for the implementation (coding, programming) of

> the simulation model due to the: (1) unavailability of a simulation programming language, (2) inapplicability of a simulation programming language for the problem domain, (3) lack of knowledge of the appropriate simulation programming languages, and (4) need for integrating the simulation model with another software system. (2:287).

All of the languages listed above have been successfully used in simulations. Appendix B provides insight into the many advantages provided by Ada and why it was chosen for the LB design and development.

2.2.6 Validation and Verification. There are four areas of concern in validating a model. They are input validity, design validity, output validity and face validity(40). Most experts agree that there are no verified or validated models of actual combat processes (9, 21, 65, 40). Further discussion of validation or verification can be found in these same articles as well as (4, 61). Actual validation and verification for gaming models used for educational purposes may not be as critical as for other simulations, so long as the educational objectives are met and the participants are provided the correct insights and guidance from the simulation.

2.3 Combat Processes

Inherent in any land combat simulation is the modeling of the basic combat processes: maneuver, attrition, and command, control and communications; those processes closest to the fundamental role of army units, which is to "shoot, move, and communicate"(61:1). Attempts are made in models to describe the combat processes in detail. The model descriptions often do not model how the combat processes actually occur and often are not based on Army doctrine. Aggregated models may homogenize the attributes and characteristics of some of the processes that are usually explicitly modeled in a high resolution model. This section provides a brief discussion of some of the principal combat processes that could be modeled in a war game simulation.

2.3.1 Missions. All tactical missions fall under the general areas of close, deep or rear operations, with the operations determining the appropriate types of mission(29). For the United States Army, operations are broken down into three general categories: offen-

sive, defensive and retrograde operations. Offensive missions include movement to contact, meeting engagement, hasty attacks, deliberate attacks, exploitation and pursuit(29). Although FM 100-5 Operations considers envelopment, turning movement, infiltration, penetration, and frontal attack as "forms of maneuver", for the purposes here, they will be considered as forms of the offense, whereby they could be implemented whenever an offensive mission is conducted by a unit.

Defensive operations include mobile and area defense; both hasty and deliberate(29). Retrograde operations include delays, withdrawals, and retirements(29). Special purpose operations include reconnaissance in force, attacks from a defensive posture, diversionary operations, offensive reliefs, and raids(29:127–128). Except for attacking and defending along a single front, few models simulate other large scale warfare operations such as sieges, deep penetrations, envelopment, exploitations, and loss of FEBA linearity(9).

Because of the many types of missions, and their similarities, missions should be specifically identified and their characteristics provided to the user prior to their use. The next few subsections provide generic characteristics and descriptions of missions that could be modeled in a war game.

2.3.1.1 Tactical Maneuver. "Maneuver is getting combat power to the right place at the right time to concentrate against an enemy weakness and defeat him"(49). Forms of maneuver are common to all offensive missions(29). For example, to attack a certain location implies a movement to that location.

Searching for enemy units is a critical portion of maneuver. This can also be considered the movement to contact portion of an offensive operation. A searching unit must detect enemy units and select which unit will be fought(35). Unit maneuver is often based on the attributes of the units, the environment, and is directly correlated to the mobility of a unit. The next three subsections briefly discuss specific operations that might involve maneuver.

2.3.1.2 Attack. Generally, a unit conducting a movement to contact will perform a hasty attack when meeting an enemy unit unless a significantly superior force is

met(29). The deliberate attack is better planned and uses the full power of the unit. Exploitation and pursuit are integral parts and extensions of the attack.

2.3.1.3 Defend. The two traditional defensive arrangements are mobile and area defenses. "Mobile defenses employ a combination of offensive, defensive and delaying action to defeat the enemy attack"(29:134). This is quite difficult to model as it requires envelopment and maneuver of mobile defensive reserves operations. "Area defense is usually conducted to deny the enemy access to specific terrain for a specified time" by absorbing enemy contact into well prepared defensive positions(29:135).

2.3.1.4 Withdraw. Retrograde operations are "movements to the rear or away from the enemy"(29:153). There are three types of retrograde operations: delays, withdrawals, and retirements. Delay operations keep a unit in contact while moving to the rear. A unit is essentially giving ground in order to gain time. Withdrawals are voluntary disengagements from enemy forces. Generally, the unit is freed for a new mission or is merely preserving its force. Retirements are conducted by forces not in combat and are merely orderly movements to the rear(29).

2.3.2 Attrition. Although it is generally true that "it is the purpose of combat to cause attrition and the measurement of the attrition on either side is output which is most used to define the results of battle", it is not necessarily true in all cases, particularly in NATO(32). Attrition is probably the most discussed and modeled area of combat modeling(61). It is used to determine battle outcomes, winners and losers. The attrition modeling process often dominates and drives all parts of a combat model(65). This is unfortunate since the real purpose of combat is "to defeat the enemy through various means, of which attrition is just one method"(49). Fortunately, attrition modeling is less important in war games as compared to other simulations. Although many ways of modeling attrition have been used including those based on pure judgment, the two most commonly used methods for generating attrition due to combat are based on Lanchester differential equations and firepower scores(47). Both methods are used deterministically in simulations(61).

The use of Lanchester equations for modeling attrition have received the most study and discussion of all the techniques used to calculate attrition. The basic premise of the equation is that the side with the greatest effectiveness attrits the other side at a higher normalized attrition rate. Its primary focus is on force-on-force operations with attrition being a function of attrition rates. The mathematical operators are difference or differential equations.

> The basic idea of the Lanchester square law is that the casualty rate over time of a force is directly proportional to the number of enemy engaged. This is modeled mathematically by equations such as:
>
> $$\frac{dA}{dt} = -E_b B$$
>
> and
>
> $$\frac{dB}{dt} = -E_A A$$
>
> where E_B is "effectiveness" of B expressed as the number of A killed per unit of B's force, and E_A is the "effectiveness" of A defined analogously(6:23).

Extensive discussion of Lanchester equation theory can be found in (3, 7, 17, 20, 31, 37, 38, 46, 55, 61, 63).

A firepower score is usually a single number which represents the entire combat capability of a unit. This represents a point of reference for a unit based on some arbitrary unit standard or the unit's own Table of Organization and Equipment (TO&E). It is usually directly correlated to the main weapons systems of the unit including small arms and crew served weapons. Firepower scores are typically based on historical tests, field data and expert judgment(47). In a very highly aggregated model, firepower scores are an aggregate of many weapons systems and are manipulated for combat ratio purposes(47). It is essentially a method for "aggregating the heterogeneous forces on a side into a single equivalent homogeneous force"(61). Recently, firepower scores have been the primary method used for modeling large-scale combat. They are also used for determining engagement outcomes, casualties, and FEBA movement(61). Additional discussion of firepower scores

can be found in (4, 44, 61). Other less common methods for modeling attrition, such as quantified judgment methods or eigenvector methods can be found in (8, 18, 17, 58, 61).

Both Lanchester equations and firepower scores can only approximate the effects of combat destruction, for neither model reality correctly and their computational complexity increases quickly as these methods attempt to more closely model what really happens on the battlefield(41, 47, 61). Additionally, disaggregation is also a problem, that is, once attrition has occurred, the scored results of the exchange must be converted to ascertain the numeric combat losses of the individual components. This is quite difficult to do mathematically and is often capricious in selection.

Attrition in typical models occurs when units are in some proximity of one another, i.e. the same grid square or when enemy units occupy adjacent grid squares. Attrition is usually broken down into attacker/defender or meeting engagement types. The defender usually gains a defender's advantage as he remains in a given location(24). Some models assess strength losses from "tactical surprise" when a unit is attacked from a direction other than its own primary direction(24). No matter how attrition is performed, it must fulfill the requirements of the simulation.

2.3.2.1 Breakpoints. This is usually a percentage of a unit's strength indicating the threshold point at which a unit engaged in combat would withdraw from combat. It is seldom the case where a unit is completely destroyed(65). This is a very important area, as breakpoints essentially determine who wins the battle in many simulations. Breakpoints can affect logistics, attrition, battle duration and equipment losses. The breakpoint is often assigned between 5% and 50%(32), 30% being the most often used(36). The breakpoint can be calculated in many ways, the three most common being listed below. The first way is based on an absolute total combat capability a unit has, the second way is based on an attrition rate being applied against a unit at a given time, and the third is based on a unit's strength relative to its original strength. Although these definitions and values are prominently used, they surely cannot be the criteria a NATO commander would use in combat even though its simple form has a nice structure. Unfortunately, it has been

> demonstrated conclusively what many thoughtful analysts have long known: a percentage casualty threshold, or breakpoint (whether it be 30 percent or any fixed percent) is a totally unrealistic determinant of combat termination in computer simulations(36:41).

In real life breakpoints do not determine who wins(49). The three most important factors in breakpoint modeling are tactics, relative combat power and assessment of casualties(36). Time step and event algorithms have been developed for determining the end of combat; however, they are often complex or difficult to implement(36). Player or controller specified breakpoints are generally sufficient for war games.

2.3.2.2 FEBA Movement. Although Forward Edge of the Battle Area (FEBA) movement is not a direct combat process, its characteristics are defined by combat processes. Two units of opposing sides in contact generally define a small portion of the FEBA, often closely related to the Forward Line of Own Troops (FLOT). The FEBA is often specified along the entire front for the theater where the two opposing forces are in contact, but it is actually "the terrain line that separates the covering force area from the main battle area"(32). The FLOT is the contact line between opposing forces. The FLOT and the FEBA are often considered the same. FLOT movement is usually based on the relative combat powers (force ratios) of the units in contact, mobility factors, force postures, and obstacles(41, 47).

Maximum rates are often specified for the FLOT movement so that unrealistic movements do not occur(18). The actual drawing of the FEBA or FLOT is desired in war games in order to give the players some sense of the mechanics of the simulation and insight into who is winning or losing. There is no implication here that smooth FLOT lines equals success. Commanders may risk negative FLOT movement in some areas for operational gain in others(49).

2.4 Battlefield Environment

The battlefield environment includes all aspects which have an effect upon combat, such as line of sight, trafficability, obstacles, electromagnetic pulse, fallout, weather, and

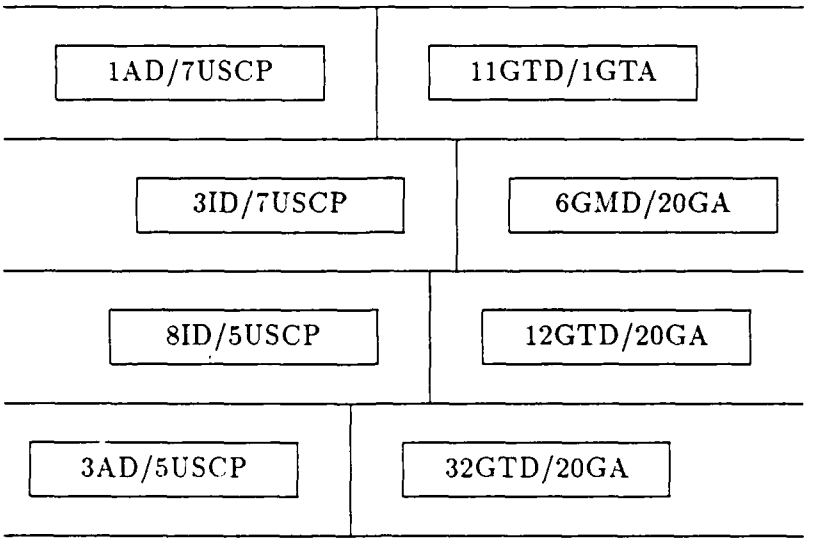

Figure 6. Sector Terrain Model Representation.

day/night effects(23:23). Some of the more important environmental aspects of terrain are discussed below.

2.4.1 Terrain and Mobility. Terrain areas occupied by units are usually defined for program control purposes. Battlefield terrain is usually subdivided into sectors, networks or grids for representing or controlling trafficability, movement, position, area targets, and aggregation required for the maneuver processes(34). Grids may be square, rectangular or hexagonal in shape.

In sectors, units typically move, attack and engage only against enemy units in opposite sectors and never cross lateral sector boundaries. The number of sectors may be limited. Programming logic is often simplified because of the restricted nature of possible unit movement. This may also be considered a disadvantage since all movement across sectors is nonexistent(52). See Figure 6 for an example of sector representation. Opposing units are on each side of the vertical lines in each horizontal sector.

The modeling of terrain using networks employs nodes and lines between nodes to represent routes between points. Precision is lost, since reality is forced or compartmentized into nodes and arcs. Terrain representation is often thought of as a compromise between

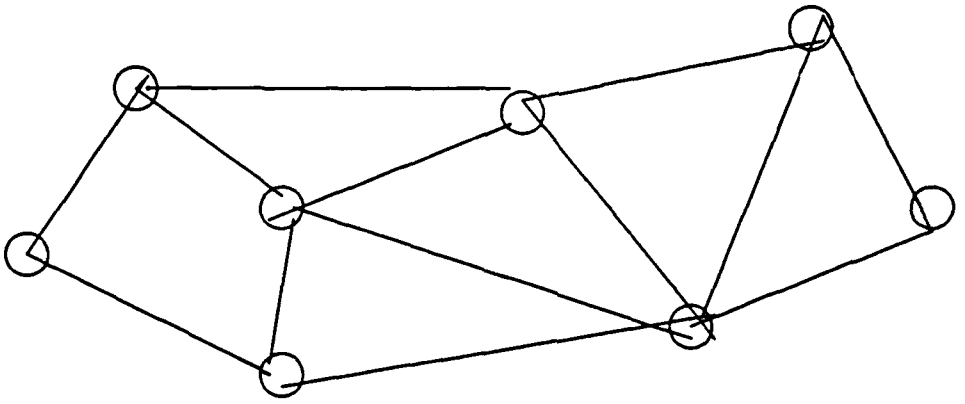

Figure 7. Network Terrain Model Representation.

the grid and sector methods of terrain representation(52). Networks are often used for logistics models. The paths and nodes can be given attributes used for logistics movement calculations(34). See Figure 7 for an example of a network configuration.

Grids defined as squares or hexagons (hexes) are probably the most prevalent form of terrain construct. Grids are fairly easy to define and are easily understood by the users. See Figure 8 for examples of a square grid configuration. Hexagonal grids provide advantages over squares in that the distances between hexagon centers are all the same distance and two additional boundaries are available for terrain characterization. This can provide additional flexibility in characterizing the terrain. Figure 9 shows a hexagonal system.

There are many methods to model terrain features and characteristics in hexagons. Depending upon the size of the grid, models often apply a given terrain attribute to an entire grid square or hexagon, usually as a method to economize. Thus, an entire grid would be considered "forest" or "mountain". This is certainly reasonable for an aggregated model. Common terrain types include highway, minor road, cross country, desert, hills, mountains, urban, swamp, and water. Terrain features and obstacles may lie in grid squares or along boundaries. Some terrain features and obstacles may degrade or completely stop movement. Some obstacles might be emplaced or removed by units in the simulation. Often, rivers run along the boundaries while roads run through the boundaries(24). An

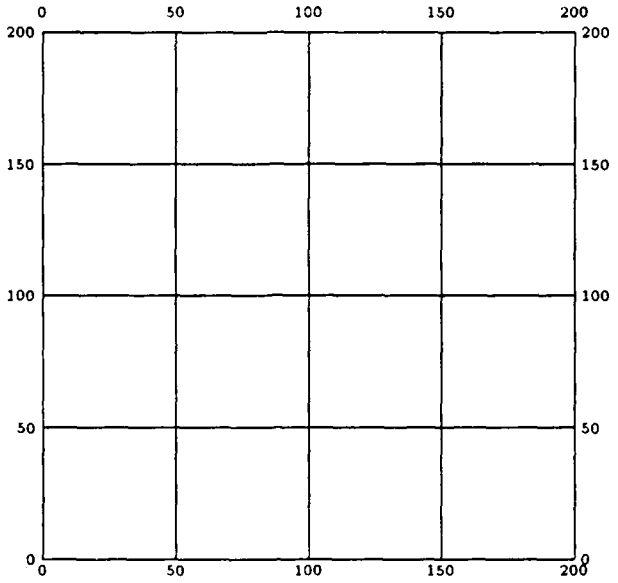

Figure 8. Square Grid Terrain Model Representation.

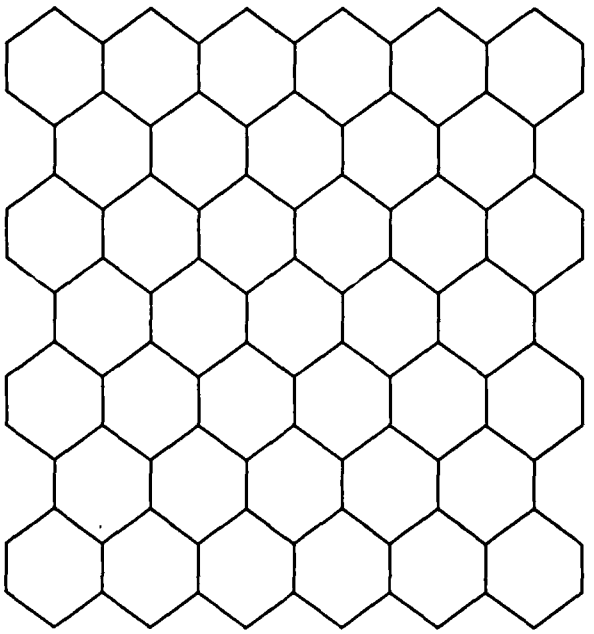

Figure 9. Hexagonal Terrain Model Representation.

excess number of units in a hexagon (stacking) is typically not allowed and thus units may be forced by the simulation to search for another grid.

Battlefield mobility is the rate at which maneuver can occur and is a function of unit characteristics and the natural environment including terrain features, weather, and whether the unit is in contact or not. Units are typically assigned an average ground speed number that must be combined with other factors to attain the actual speed through a grid. An example of how a model can represent the effects of attributes on mobility is(24):

Mobility(speed)= basic movement rate × terrain multiplier × movement rate multiplier × mobility degradation × weather multiplier.

Although movement algorithms should generally model reality, simple algorithms are often sufficient for war game simulations.

2.4.2 Night Operations and Weather. Operations conducted at night are usually degraded as compared to day operations. This is the case for movement, attrition, and support. Weather often has various degrees of severity. All too often, its only effect is on unit movement rates. Weather may be applied to the entire simulation or to individual grid squares. A compromise is for a unit to acquire its "weather" value from its initial position and maintain that value until it reaches its destination. Any simple means for applying night operations and weather to combat processes is usually sufficient for war games.

2.5 Combat Arms

Combat arms includes all units that directly engage in combat. Of primary interest here are armor, infantry, cavalry, aviation, field artillery, air defense artillery, air force and special operations forces units. The armor, infantry and cavalry divisions will be the units most likely engaged in direct close operations combat. Field artillery, air defense, and air support provide direct and general support. Combat arms operations are the most widely modeled aspects of simulations, particularly war game simulations that focus on combat attrition.

2.6 *Combat Support, Service Support and Reserves*

Combat support and service support are provided by non-combat units. Combat support is a combat multiplier and is represented by engineers, military intelligence, signal and communications, military police, etc. Combat Support units usually do not engage in direct conflict with the enemy forces.

Combat service support units include finance, ordnance, quartermaster, transportation and any non-combat activities necessary to support battles(29). This encompasses all forms of supply and logistics and is essential for sustained combat and the extended survivability of a unit. This includes Army categories of supply played in models including petroleum, oil, lubricants, ammunition, repair parts, nuclear, chemical, food and others. Medical, construction services, graves registration operations, maintenance, transportation and field services are also often represented(23).

Logistics can be quite difficult to model and must include the movement of all materiel from rear echelons to the forward forces. Consumption is based on activities, relative combat strengths, average usage rates and stockage levels. Additionally, logistics is often only used as a degradation factor to a unit's combat effectiveness. In short scenarios it may not be modeled. This is particularly true for a "come as you are war". For a short war scenario (i.e. 5–10 days), using only the prepositioned war material stockage might be considered sufficient for modeling purposes.

Reserves may be played explicitly, implicitly or not at all. Players may be provided with reserves to insert into operations where necessary or every unit may be periodically reinforced with reserves. Reserves are important when modeling rear operations(49).

2.7 *Command, Control, Communications, and Intelligence (C^3I)*

C^3I is perhaps the most difficult aspect of combat to model. Modeling of C^3 must necessarily include communications units, electronic warfare units, communications lines and electronic counter measures representation at all echelons. Planning, operations and message processing must also be considered from the lowest unit to the highest unit. Primary and alternate command posts at all levels of command must be modeled(23).

In war games, when it is modeled, C^3 is either automated or involves player interaction. Automated simulations are quite difficult to develop and usually require extensive "if-then" constructs. There is a growing trend towards increased use of expert systems and artificial intelligence in the modeling of command, control and communications. Encompassed in this development are further enhancements in the areas of intelligence, target acquisition, and electronic warfare(40). Few models represent decision making processes well, especially large scale models(9). Training war games usually model C^3 only through the player interaction, exclusive of the actual computer simulation run.

"Combat intelligence is that knowledge of the enemy, weather and geographical features required by a commander in the planning and conduct of combat operations"(26:2-1). This area is often not modeled in simulations, particularly because of the difficulty of quantifying it for simulation purposes. When properly modeled, intelligence play must include the four phase intelligence cycle: directing the collection effort, collecting the information, processing the collected information, and disseminating and using the resulting intelligence(26:2-15). These operations are continuous in nature and usually take place concurrently. Other intelligence areas commonly modeled include deception, electronic warfare and jamming, and operations security effects. Databases of information must usually be maintained(23). This area may require exceptional overhead and its overall importance for a war game probably does not require full scale modeling. In war games, actual player and controller interaction may provide a reasonable semblance of intelligence play, whereby players are given intelligence data from the game controllers. Indices in the game may be used to filter enemy information provided to players. This information is only a partial information of the opposing sides status of forces. Complex intelligence play is often not necessary for a war game, but it may be important to teach the player the need and critical nature of good intelligence data acquisition and interpretation.

2.8 Nuclear, Biological, Chemical (NBC)

The degree of NBC operations continues to gain greater resolution in combat models(32). Its importance is scenario driven and can be extremely difficult to model correctly, particularly in the cases of chemical and weather interactions, delivery methods and dispersion

patterns. Contamination may be applied directly to degrade units, restrict use of equipment, change the tempo of battle, affect the terrain, or affect air operations(32). War games may model NBC depending on the focus of the gaming objectives.

2.9 Conclusion

This chapter presented a quick overview of some of the more important aspects of combat modeling. It is of necessity brief and certainly not all encompassing. Many of the topics discussed may not be required for interactive war games or simulations. The most important aspects of each area were discussed to form the basis for the requirements specifications as they will be applied to the LB. This is presented in Chapter III.

III. Initial Land Battle Specifications

3.1 The Land Battle in General

This chapter provides a general discussion of land combat modeling as it applies to the specifications and requirements for the Land Battle (LB) program. This discussion provides only the requirements specifications for the program development. Although the initial focus of the Theater War Exercise was war in the NATO arena, the LB program was designed to be completely generic and applicable to any combat arena where the focus is on units and their support relations.

Since the LB is an aggregated model, the principal area of focus is not on the subordinate units below division level, but rather on corps division and non-divisional units (including corps regiments and combat support brigades) to army group level. It was chosen to be a stochastic model in that player decisions form the basis of the inputs.

Discrete events using fixed time steps are used to control the events that might occur. This can be done since the LB is aggregated to division level and time steps of sufficiently small size were used. A very reasonable compromise was for the time step to occur every three to eight hours. All operations and processes for every unit are performed in each time step as necessary.

Flat files were chosen as the source of input for data. This provides the most flexibility for data input and output among many machines. All information is read into the simulation from the files prior to a simulation run and read out to the files upon completion of each run. The Oracle data base management system's tools was chosen for the production of the flat files. Information for the files are produced by the database management system as a result of the user's inputs into a screen based graphics front end. The files are the only common link, other than the player, between the air and land simulations. Although most of the data required by the LB is not associated with the air portion of the exercise, the LB program must read files that contain information necessary for the air-to-land interactions in the simulation. This includes RECCE, BAI and CAS interactions.

3.2 Land Battle Combat Process Specifics

Discussed below are key combat processes necessary in the LB program. They specifically include missions, fire and general support processes, and all aspects of unit movements through terrain and weather. Breakpoints and Forward Line of Troops (FLOT) movement are two items interrelated with combat processes that are also discussed.

3.2.1 Missions. Missions include all the directed actions that a unit may receive during the simulation. Since all missions can be categorized into three general types and only general missions would be provided to subordinate units by the army group, the LB only needs to model attack, defend and withdraw missions. These missions are in line with the general directives expected of an army group commander and staff. Maneuver and the three missions as they apply to the LB are discussed below.

For the purposes of the LB, tactical maneuver includes all movement operations implied in a mission. Envelopment, turning movement, infiltration, penetration, and frontal attack are not explicitly modeled in the LB, but are allowed through player input. The LB should allow only the final mission destination to be provided to a unit, thus specific routes are not provided to the simulation unless designated every cycle. The LB must use search algorithms to select optimal movement paths from hex center to hex center across hex boundaries. Movement rates are a function of unit type and terrain attributes. Terrain and weather attributes are applied to the entire hex area. Movement is performed for a specified time period. While a unit is moving, a search algorithm is used to determine if a detected unit should be engaged. All of this simulates those actions that commanders of units below the army group level would be expected to make.

In the LB, all phases of offensive operations are simulated by the attack mission. Attack specifies that a unit moves to a general terrain location (a hex at least 10 kilometers across) and that it engages with equal firepower enemy units in adjacent hexes along its direction of movement. Movement is performed as long as no enemy contact occurs. If enemy contact occurs, attrition occurs. No special offensive operations are modeled.

Area defense is the basic form of defense modeled in the LB. All types of defend operations are simulated by the defend mission. Defend specifies that a unit is to move to

a specified area and remain there. A defender unit's combat capability or fire powerscore are increased as his length of stay in his location continues. Two opposing units in adjacent hexes, both with defend missions, do not attrit one another. This is what would be expected of actual units in contact.

3.2.2 Attrition. The LB must assess attrition when opposing units meet in adjacent hexes. Although attrition could be modeled with units occupying the same hex, control of this type simulation is more difficult and not nearly so clearly defined as compared to when opposing forces are not allowed to occupy the same hex. Defender's advantages are considered and applied to unit's strengths. Firepower scores are used for every unit engagement making computer calculations faster and easier, and providing the player with a quicker and keener grasp of the capabilities of a unit. Although firepower scores can be used to define a unit's component combat losses (such as percentage tanks, armored vehicles or personnel), this reduction method is not often realistic or credible. These results would probably not be of immediate importance to the Army Group Staff. Firepower scores are the sole indicators of combat power for the units and no disaggregation of firepower is used. Attrition is based on force ratios, engagement type, unit posture (attacker or defender), and terrain characteristics.

Player specified breakpoint percentages are also used. Hex boundaries can be used to mark FLOT lines when opposing units are in contact.

3.2.3 Terrain and Mobility. Interlocking hexagons were chosen for use in the LB. Calculations for hex movement are well defined and easily understood. The LB receives the attributes of terrain features and obstacles from the database, assigning those attributes to entire hexagons and boundaries between hexagons. Thus the terrain hexes explicitly model unit mobility possibilities. Obstacles are emplaced and removed by units in the simulations or by the players directly manipulating the database before simulation runs. Trafficability is represented in the hex and at the boundaries of hexes. Boundaries contain bridges, mines and manmade obstacles which can be created or destroyed by engineer, fire support and air force units. Units at boundaries search all adjacent hexes in the direction of movement for obstacles, terrain features and enemy units prior to movement. Impass

terrain features cannot be traversed by a unit. This provides the army group staff the opportunity to select and destroy interdiction targets.

Weather attributes are assigned to hexes as appropriate and consist of six levels of severity. The weather attributes are assigned by the weather officer at army group during interactive play prior to the simulation run or by acquisition of the data from the air portion of the simulation.

3.2.4 Combat Arms Operations. The LB must play infantry, armor and cavalry divisions; artillery and air defense artillery brigades; reconnaissance, air interdiction and close air support. Division and corps non-divisional brigades are explicitly modeled in the LB, while air support may be applied against units and targets. Fire support may be provided on request to units if the fire support unit has been designated as direct support to the requesting unit. Thus both simple general support and direct support operations are modeled. Players indicate which combat units are supported by combat support units. The allocation of land operational fires could also be expected of the army group, even though theater air forces are the primary source(67). All air force support data (available sorties) is provided from the air portion of the Theater War Exercise for application to units requesting that support. Air interdiction, battle air interdiction and close air support is modeled directly.

Although all Combat Support and Combat Service Support units and operations are important to combat, in the interest of modeling simplicity, the LB should explicitly models engineer support. The LB allows reserves (units and unit firepower) which are under the direction of the army group. Reserves can be applied as resupply support or as units to be used in combat.

3.2.5 Command, Control, Communications, and Intelligence. The LB models command, control and communications only through player interaction prior to the actual simulation run and through the intelligence gathering and dissemination process.

3.2.6 NBC. NBC is not critically important at this time for the LB portion of the Theater War Exercise and so because of time constraints, is not modeled.

3.3 Modifications to Initial Design

During the LB development process, from both the ongoing discussions with the Air Force Wargaming center and the acquisition of new wargaming knowledge, several improvements were made to the initial model design. These changes have been incorporated into the model and are mentioned here for completeness. Primary reasons for changes in the original design were based on a perceived requirement for additional LB capabilities, necessary additions of processes to enhance the encapsulation of AirLand Battle doctrine, and the existence of sufficient time to make those changes. The most significant changes occurred in three phases. Phase I encompassed the development of maneuver, attrition, engineer and field artillery support. The overall program was set up, initial program flow was determined, and the most primitive data structures were encapsulated in code.

Phase II additions included:

- Addition of aviation and cavalry units as unit types.
- Addition of logistics as an integral part of the simulation with effects related to attrition and maneuver. Support and depot units were added to model depot, logistics and resupply movement and operations. Logistics units have a SPT (support) mission. Ammunition, petroleum products and general replacement supplies are the only classes of supply modeled. This was a major criticism of the original TWX game.
- Addition of retrograde operations. All retrograde operations are implicitly simulated by the delay mission and explicitly modeled by the withdraw mission. The simulation automatically gives a delay mission to units that sustain an unreasonably high combat attrition rate or reach the combat breakpoint specified by the player. These units disengage from combat and move towards their rear. A player explicitly issuing a withdraw mission directs a unit to move to a rear location specified by the player. This means a direct and immediate separation from combat.
- Initial encoding of the BAI and CAS air force support.

Phase III additions encompassed the following:

- Addition of intelligence as an integral part of the player interface and simulation. Intelligence play required the addition of military intelligence brigades as assignable assets. Special operations forces were added as units to apply against general locations for intelligence gathering purposes. The modeling of corps and divisional intelligence gathering assets was added. These assets modified the intelligence index of units they are applied against. The LB uses an intelligence index as an attribute of each unit. This index indicates the amount of partial information (type of unit, location, combat power, etc.) to be provided to enemy forces. Intelligence indices are controlled by tactical reconnaissance forces from the air portion of the simulation and war game controllers. Special operations forces, division and corps intelligence gathering, and military intelligence brigades provide land intelligence information.
- Addition of movement to contact (move) as a mission type. This allowed a unit faster movement than the attack mission.
- Addition of different movement and attrition rates for each unit based on unit type, terrain, and mission. Original movement rates were based only on a unit's type and terrain.
- Modification of the all program rates so that they would be based on a single day. This gave the controller the option of specifying any number of time slices per day and allowing the simulation to run for any number of days. Other changes involved the movement of all the previously hard coded constants from the simulation to a constants file that could be easily modified by the controller.

Phase IV modifications focused on the integration of the LB program with the current Theater War Exercise. The most important changes included:

- Addition of interdiction as an attribute of terrain obstacles and units. TWX Air requires this for differentiation between BAI and AI. All targets beyond a certain distance from the friendly FLOT are considered to be interdiction targets.
- Addition of surface-to-air (SAI) indexes for units and terrain. TWX Air requires this for determining how many surviving RECCE, CAS and BAI sorties will be applied

against the land targets. All units are given an inherent SAI index based on their type. Terrain hexes receive an SAI index based on the units occupying the hex.

- Addition of a delay to moving units as a result of BAI being applied against them.
- Modification of the attrition alogrithm so that attrition is applied proportionally, rather than equally.
- Addition of an intelligence filter based on the intelligence index. The intelligence reporting algorithm was changed to more correctly reflect reality.
- Addition of day and night cycles for application of BAI, CAS, and RECCE sorties.

Phase V changes involved the necessary modifications to the program to work under the constraints of an IBM PC based system and integration with the Oracle database management system. Problem areas included the 64k data segment limit and the 640k RAM limit. These problems are discussed in Appendix E.

These changes added considerable capability and flexibility to the program.

3.4 Conclusion

Based on the requirements specifications above, the LB program contained the following processes:

- Data input.
- Data control structures.
- Subprograms for:
  - Initialization.
  - Assessing unit firepower and combat powers.
  - Assessing unit contacts.
  - Direct and General Fire support.
  - Engineer support.
  - Air Force Fire support (CAS, BAI).

  - Intelligence play (Army, Air Force)
  - Attrition.
  - Maneuver.
  - Support and logistics.
- Data output.
- Report writing.

Chapter IV provides an object oriented preliminary design for the LB based on the requirements presented in this chapter, with a focus on all the processes just described.

IV. Initial Design of the Land Battle

The purpose of this chapter is to detail the initial design of the land battle (LB) program using object oriented design (OOD) methods. Booch and the EVB Software Engineering manual are the primary sources for the development effort(11, 30).

Essentially, OOD "is a method that lets us map our abstractions of the real world directly to the architecture of our solutions"(11:47). OOD concentrates on the design and implementation phase of software development, while still allowing other software development methodologies to be used. OOD philosophy, its place in software development and its primary benefits are not discussed. Further information can be found in (1, 10, 11, 19, 30). The major steps for OOD espoused by Booch and which are used in this thesis are (11:48-51), (30:2-1):

1. Define the problem.
    - Analyze and clarify the givens.
2. Develop an informal strategy for the problem domain.
3. Formalize the strategy by:
    (a) Identifying the objects and their attributes.
    (b) Identifying the operations that affect each object and the operations that each object must initiate.
    (c) Grouping operations, objects and types.
    (d) Establishing the visibility of each object in relation to other objects.
    (e) Establishing the interface of each object and the operations.
4. Implement each object and the operations.

The first three major steps are considered as part of the design effort. Implementing the objects and operations encompasses actual code development. Application of these steps are used in the development of the land battle program and generate the necessary

operations and objects to actually solve the problem. Further discussion of OOD are provided with each step of the process.

4.1 Defining the Problem

Before the major substeps for OOD can be applied, a general understanding of the problem is required. This is typically done by stating the problem using one or two sentences, which are elaborated upon in successive steps in defining the problem. The problem statement from Chapter I is used as a basis for this effort. For this design, a statement of the problem is:

> Develop a land battle program to interface with the Air portion of the Theater War Exercise that simulates theater level land combat.

This sentence provides a starting point for initial understanding of the problem necessary to start the development process.

4.1.1 Analysis and Clarifications of the Givens. In this phase, all pertinent information is examined for possible limitations bounding the domain of the solution. The primary effort here is clarifying the problem through constraint identification. The most important criterion here is that all information (from the perspective of the actual program user) has been found that is absolutely necessary to solve the problem. Anything not specified may be solved at the discretion of the implementer(30:3-11). Deliverables from this phase are the necessary information required for problem solution(30:3-13). Chapter III provided the basic requirements specifications for the LB development. Additional givens for the land battle program based on Air University constraints and previously discussed assumptions include:

- The Land Battle (LB) program had to be capable of running on a multitude of systems including: MicroVax III, Vax 8650, SUN 386i, and Zenith Z-158 personal computers.

- The LB was written so that necessary data was accessed from flat files. 4th Generation Language database tools are used for some input, output and report generation. Data is acquired from the Air portion of the Theater War Exercise.
- The LB must be capable of interactive user play or as preprogrammed play with minimal player interaction. This necessitated preset inputs for the program, so that player interaction was not initially required.
- Turnaround for the simulation run had to be under three hours for a 24 hour simulated combat cycle.
- The LB is viewed from the army group level of play with corps major non-divisional units as the lowest level of aggregation.
- The LB must be interfaced with the Air portion of TWX.

With these additional constraints on the development of the program, an informal strategy can be developed.

4.2 Developing an Informal Strategy

A single paragraph is the deliverable for this step of the design process. The paragraph must be detailed, but generally not longer than nine lines. Essentially, the LB must represent the general tactical aspects of land warfare based on player inputs. This paragraph is a general solution to the problem and is often considered a "plan of attack"(30:3-18).

> A unit receives orders and performs required missions based on the content of the orders. When a unit comes into contact with an enemy unit it may receive land and air fire support. Obstacles are installed and destroyed by engineer units. A unit in contact with an enemy unit experiences mutual attrition when one unit is attacking. Opposing forces in contact form the basis for the Forward Line of Troops (FLOT). Movement is based on terrain, weather and other conditions. Reports are provided to players at the end of the simulation based on intelligence acquired during the simulation. Unit actions are constrained by logistics resupply. Air Force BAI, CAS, and RECCE support is provided to land units based on player inputs and the results from the air portion of TWX.

This paragraph is the basis for the formalization of the strategy and represents what must be implemented by the solution. Because the development of a land battle program is so large, the paragraph is descriptive rather than algorithmic. Further detail is provided in successive steps.

4.3 Formalizing the Strategy

This phase involves a detailed analysis of the informal solution presented above. It consists of four substeps which break the solution down into detail. This is where the formal process of design starts to take effect. All objects, operations, visibility and interfaces are examined in detail. This step provides the first major look into the true structure of the final product, while providing necessary checks for a solid, cohesive design. For each phase, different levels of abstraction are examined.

4.3.1 Identify the Objects and Their Attributes. The first substep of Formalizing the Strategy involves identifying all of the objects that can change state or can cause some type of action to occur. Similar type objects may be grouped together into classes or types. Nouns from the specification or functional description of the program are used as a basis for the objects. At the highest levels of abstraction are the following major objects or object classes.

- UNITS. The basic operational objects manipulated by the players.
- GRID_TYPE. The data structure for terrain representation.

DATA includes all data manipulated by the program including constants and attributes of the other major objects. Discussion of these objects and their attributes can be found in Appendix C. Once all objects of interest have been identified, all necessary operations that manipulate the objects are identified.

4.3.2 Identify the Operations and Their Attributes. This section "identifies the operations that affect each object and the operations that each object must initiate"(11:48). Generally, any operation specified here will operate on only one of the objects specified

previously. The descriptive definition paragraph is used as the basis for identifying operations. Generally, verbs and verb phrases provide the operations. Attributes of the operations are discussed in general detail. The highest level of abstraction includes the following operations:

- DATA_INPUT: This occurs under INITIALIZE and is one of the first operations performed under the LB. It moves data from the database into appropriate data structures of the LB program. All objects have attributes represented by database elements. Required attributes are also initialized.
- DATA_OUTPUT: One of the last operations performed by the LB program. Current data from the LB is written to the appropriate databases. This occurs in WRITE_DATA.
- REPORT_WRITER: Writes all of the reports for the player's use. May use data directly from the database or acquire data via the relational database system. This occurs after every simulation run. Necessary reports are sent to the printer. Filtering for intelligence reports is performed based on the INTEL_INDEX for any given unit.

The lower level of abstraction focuses on the operations manipulating the objects of this same level. The operations included here are specifically those under the control of the main procedure. The main procedure control is based on user inputs. A value for NUMBER_TS_PER_DAY indicates the number of time slices a single day will be broken up into. This specifies the actual length of the time step. TOTAL_NUMBER_OF_TS provides the actual number of time periods the simulation will run. This indicates the number of days in the simulation. Main procedure calls with subordinate calls include:

- INITIALIZE.
  - GET_CONSTANTS. Reads all necessary constants and user specified values necessary to run the simulation.
  - GET_GRID. Gets the grid hex data from a file.
  - GET_UNIT. Gets regular land unit and logistics unit data from files.

- GET_SF_INTEL. Gets the special operations forces data from a file.
- SET_UP. Sets up the initial hex and unit relations. The third level of abstraction controls a unit's connections to the hexes. ADD adds a unit to the hex and DELETE deletes the unit from a hex.

- GET_AF. Controls data input of air force data.
  - GET_RECCE. Gets air reconnaissance data from a file.
  - GET_CAS. Gets close air support data from a file. linked list.
  - GET_BAI. Gets battle air interdiction data from a file.

- LOG_SPT. Controls basic depot logistics resupply operations.
  - DEPOT_LOG. Operations to resupply depots with supplies.
  - UNIT_LOG_SUPPLY. Operations to resupply land units with POL, AMMO and HARDWARE.

- SET_UP. Performs necessary operations to set up for combat and attrition operations. Includes ASSESS_CP (assess combat power), SET_ATK, and ASSESS_CONTACT. This is performed when two opposing units are in contact and must be true for two ground combat forces to attrit one another.
- APPLY_AFS. Applies BAI and CAS to land units through APPLY_BAI and APPLY_CAS subprograms.
- ATTRITION. Performs the overall control for combat attrition operations.
  - APPLY_FS. This provides fire support and includes aerial, aviation, field artillery and air defense artillery.
  - APPLY_CP. Distributes combat power to all adjacent units in attrition with this unit.
  - ATTRIT. This performs the attrition calculations on the firepower scores of the units.

- DESTROY. This removes a unit from the simulation when the unit is attrited a sufficient amount.
- WITHDRAW_UNIT. Separates a unit from combat and moves that unit to the rear.

- MOVEMENT. MOVEMENT encompasses MANEUVER which includes:
  - DETERMINE_ ROUTE. Determination of possible routes of movement based on initial and final destinations.
  - SELECT_ROUTE. A selection of the best of several possible routes.
  - MOVE_IN_GRID. The control of unit movement (time remaining) in grids and at obstacles.
  - BORDER_TRANSITION. Control of units as they cross hex borders.
  - UPDATE_LOCATION. Updates a unit's location in a hex.
  - OVERCOME_OBSTACLE. Overcomes an obstacle in the path of a moving unit.
- INTELLIGENCE. Controls the calculation of INTEL indexes. It includes the loss of intelligence through time (REDUCE_INTEL), intelligence gathering efforts in front line units at division level, inluding the corps military intelligence brigade (ARMY_INTEL), and directed special operations forces intelligence gathering efforts (SF_INTEL).
- AF_INTEL. Uses air force RECCE assets to modify INTEL indexes of land units.
- REPORT_WRITER. Is used for troubleshooting purposes. It writes all key data to standard output for further analysis.
- WRITE_DATA. This is used to write the data from the program back to files for later use. WR_GRID writes the hex data, WR_UNIT applies to units, WR_INTEL provides intelligence reports, and WR_DESTROYED writes the destroyed units to a file. WR_REPORTS writes the final user information reports for the player's use.

These operations provide only a general overview of more specific operations that occur under their purview. Their characteristics for the initial design process can be found

in more detail in Appendix D. The actual final implementation is discussed more completely in Chapter V.

4.3.3 Grouping Operations, Objects and Types. In this section, objects and operations are grouped so that a cohesive structure is developed. A mapping is made between operations and the objects upon which the operations act. Objects and their respective operations at the first level of abstraction are:

- DATA: DATA_INPUT and DATA_OUTPUT must occur between the databases and the land battle. REPORT_WRITER provides required reports for players. Filtered Intelligence is provided based on unit's INTEL_INDEX.

The second layer of abstraction encompasses:

- UNITS: Units are located in a GRID_TYPE. Units given a specific order to move to a new location must MOVE_IN_GRID to that location by the most expeditious manner. A unit must DETERMINE_ROUTE and SELECT_ROUTE before movement commences. Units UPDATE_LOCATION in the hexes of the GRID_TYPE as they MOVE_IN_GRID from hex to hex. If obstacles are encounterd and the unit wishes to continue on this path, the unit must OVERCOME_OBSTACLE. If an attacking unit comes into the proximity (adjacent hex) of an enemy unit it is IN_CONTACT and IN_ATTRITION. Defending units engage attacking units. Units IN_ATTRITION may receive fire support. Some units PROVIDE_FIRE_SPT. The air force will PROVIDE_AFS (provide air force support). ATTRITION occurs when units are in attrition or when fire support is delivered upon a unit. Units request logistics when supplies get low and may receive logistics support if they are available and the player has indicated for it to occur.

4.3.4 Establish the visibility of each object in relation to other objects. Visibility between the packages of the program, where the objects are encapsulated, can be found in the package specifications of the program.

4.3.5 Establish the interface of each object and the operations. In this final step of the design process all of the objects, operations and types are encapsulated and grouped into appropriate program units. All program interfaces are shown. This step gives the first indication of the program structure and topology(30). A direct result of this step is complete package specifications.

4.4 Conclusion

This chapter provided the initial LB program development based on the requirements specifications of Chapter III, the Air University, and the Air Force Institute of Technology. All objects, general operations and interfaces were specified. Chapter V provides discussion of the detailed design and low level implementation of the specifications from this chapter using Ada.

V. *Detailed Design and Implementation of the LB*

The purpose of this chapter is to discuss specific details required for the detailed design and implementation of the Land Battle (LB) program. Presentation is based on the actual flow of events in a single simulation run. The initial design from Chapter IV is used as the basis for this development. Discussion covers the data structures and program combat processes. Information about program control, data input, data output, report writing, and data formats can be found in the User's Manual.

5.1 Information Structures

The two primary information structures used in the LB program are the unit and terrain structures. The basic configuration of the objects for the LB program are arrays of records and access (pointer) types. BAI, CAS, and RECCE records are contained in linked lists. The lowest level groupings of units is composed of arrays of records containing all unit attributes (except support information) and access types to nodes that contain identifying unit information and support information. The terrain hex system is also a large two dimensional array of records and access types to nodes that contain basic unit and support information. Although access types were used initially for units, arrays were used in the final program because of the inability of MS-DOS and the Janus Ada compiler to handle access types that used over 64k of RAM. The hexes have access pointers to the nodes which are located within them. The records provide the general characteristics of the hexes, the boundaries, and attrition factors. The units in a hex at any given time are connected via a linked list of nodes. Although additional code was necessary to maintain the linked lists and access types, much additional power was gained by the capabilities to access the units from the terrain hex in which they were located. It is realized that this data implementation is not ideal and that the indirect connection between access types and array types causes some overhead. The problem could not be avoided because of Ada's type checking and the constraints of MS-DOS and the Ada compiler.

Limitations of MS-DOS caused considerable difficulty for the LB implementation that is used on the IBM PC compatible computers. Appendix E discusses these problems in greater detail.

5.1.1 Unit Structures. The package UNIT contains the specifications for all global unit objects used in the LB program. Unit attributes can be found in Appendix C. All necessary information is contained in the unit records for performing necessary calculations on units in the program. Support units contain pointers to supported units with appropriate support amounts. Although command relationships are not explicitly portrayed in code except through the CORPS_ID attribute, the players should understand the actual command relationships and some relationships are portrayed in the support attributes. BAI, CAS and RECCE records contain sufficient information for application of those assets against land units.

5.1.2 Terrain and Hex Structures. Although there are many established ways to number hexagons in a hex system, the desire for the LB was for an identical positional number system for every hex. This is typically represented by simple x-y or latitude-longitude numbering systems. Numbering for the hexes is from left to right and from bottom to top. Hexes are oriented with east-west directions across the flats and north-south directions across the points. North to south lines are at an angle of 60 degrees to the right from vertical. With this setup, movement from any hex to another hex along any direction, requires identical numerical calculations for every hex in the system. Figure 10 shows the basic hexagonal hex setup with its numbering scheme.

Every hex has internal trafficability assigned from hex center to each hex side and obstacle attributes assigned from the six sides to the center. This is shown in Figure 11. A weather attribute is assigned to the entire hex. Although unit stacking is often not allowed, an unlimited number of combat units may be maintained in each hex in this simulation. This is due to the linked list data structure used by the hexes to maintain units in them. It is incumbent upon the players to minimize hex stacking. Trafficability, weather, and obstacle values include excellent (EXC), very good (VG), good (GD), fair (FAIR), poor (POOR), and very poor (VP).

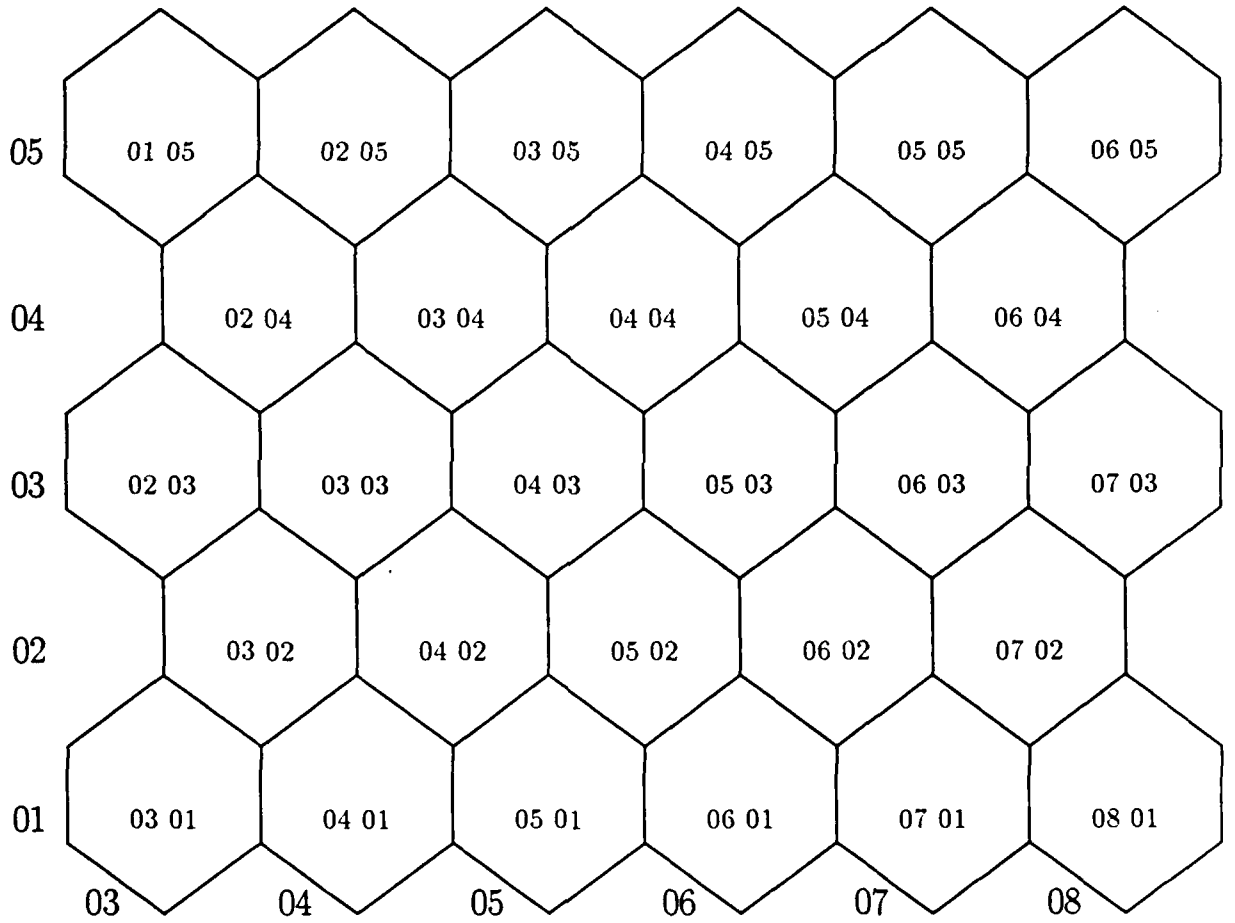

Figure 10. Terrain Grid System.

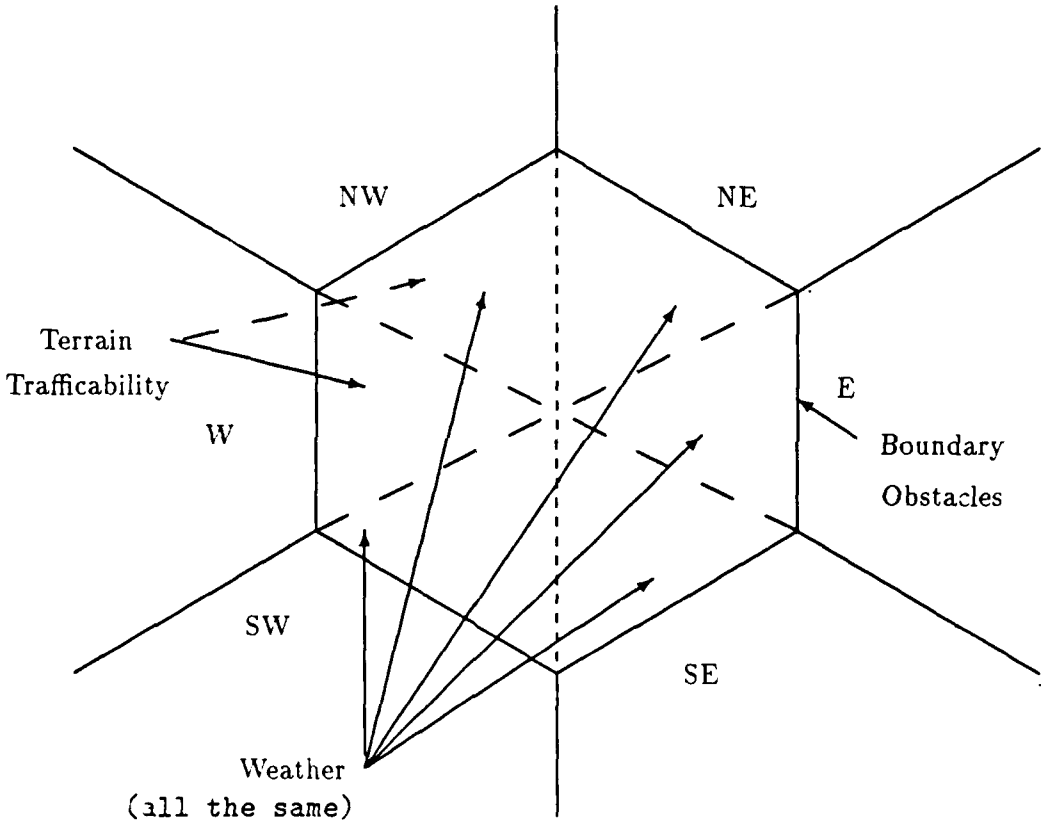

Figure 11. Internal Hex Representation.

5.2 *Maneuver*

This section discusses unit movement assumptions, constraints, and algorithms in relation to the hex system. Route determination, route selection, movement, and border transitions in a hex are the primary areas of focus. These procedures for unit movement are found in the package UNIT_OPS. Maneuver only occurs when units are not IN_CONTACT, if the unit is being attrited sufficiently to acquire a temporary implicit DELAY mission, or if a unit is explicitly given a WITHDRAW mission by the players. The procedure MANEUVER controls unit movement.

5.2.1 Route Determination. Procedure DETERMINE_ROUTE determines which of the possible six directions are allowed for unit movement. Units moving from one hex to another must have some means for identifying which axis may be moved to. The six possible axes (representing the six sides of a hexagon) are northeast (NE), east (E), southeast (SE), southwest (SW), west (W), and northwest (NW). As the distance between a unit's start and finish hex increases, the possible number of paths also increases. It is very computer intensive to determine the single optimal route between any given points. This is particularly true in a changing environment. In order to minimize all the possible axes for movement, a few constraints were necessary:

- The initial direction would be in the general axis of advance. If this is not true then the first hex to hex transition may be incorrect due to unit momentum in that direction. This is self correcting after the first hex transition.
- All unit movement must be in a forward direction towards the mission destination. Movement may not be made backwards along a movement path unless a new final mission coordinate is provided. The only exception is when a unit is automatically given a DELAY mission while in combat because of an unacceptably high attrition rate against it. The actual movement possibilities can be visualized by imagining a cone bounded by the two diagonals in the direction of desired movement from the adjacent hex along the axis of movement. Figure 12 shows the areas that would allow a movement into the NE hex from the START position given a mission hex in

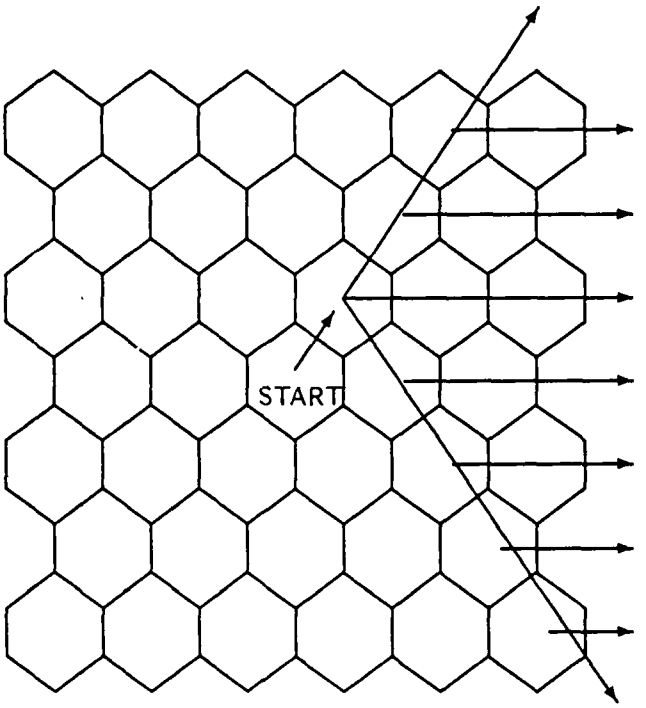

Figure 12. Examples of Possibilities for NE Hex Movements based on Eastern Mission.

an easterly direction. Figure 13 shows allowed movement into the NE hex given a mission towards the north.

Once the possible allowed directions are determined, the best route is selected from those alternatives in SELECT_ROUTE.

5.2.2 Route Selection. Procedure SELECT_ROUTE chooses the optimum route based on the allowed directions determined by DETERMINE_ROUTE. To allow reasonably quick LB execution times, several constraints for route selection are necessary:

- Only the directions allowed by DETERMINE_ROUTE are checked for possible selection.
- Route selection will be based on analysis of the present hex and only the immediate next adjacent hex in the allowed directions of movement. This is reasonable given a hex size ranging from ten to fifty kilometers. Twenty five kilometers is used in the present configuration.
- The following attributes are used to determine the optimal path:

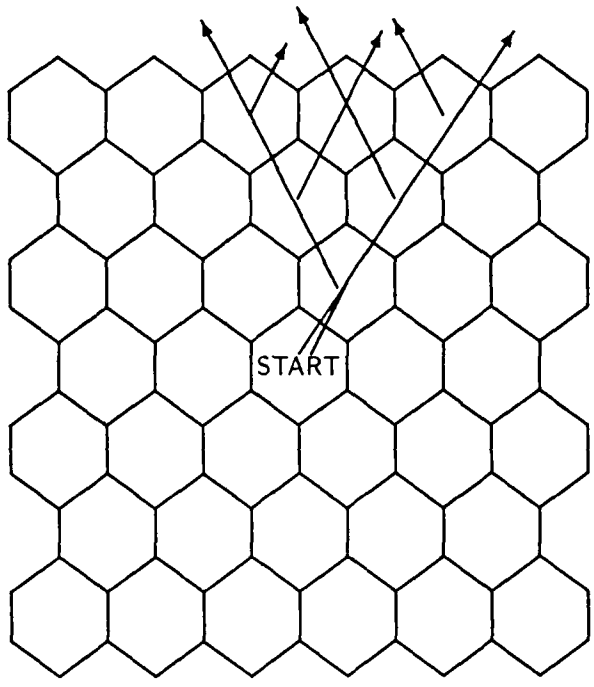

Figure 13. Examples of Possibilities for NE Hex Movements based on Northern Mission.

1. the present hex trafficability.
2. the border obstacles along the direction of movement being analyzed.
3. the next adjacent hexes' trafficability from the adjoining border to the adjacent border center of hex.
4. the weather of the adjacent hex.

Although the occupancy of hexes adjacent to the proposed hex by opposing forces units and the stacking of units in friendly hexes was considered, there was insufficient time to implement these considerations.

Routes are also weighted based on the straight line route from a unit's start to finish location. A direction that is allowed, but that is not in the general straight line path is negatively weighted. This takes into account extra distance a unit might have to move even if the terrain attributes are substantially better in the out-of-line path. Two weights are used for EW movement: when the path is above or below the horizontal parallel of a unit, and when the actual final destination hex is the next hex to the immediate E or W.

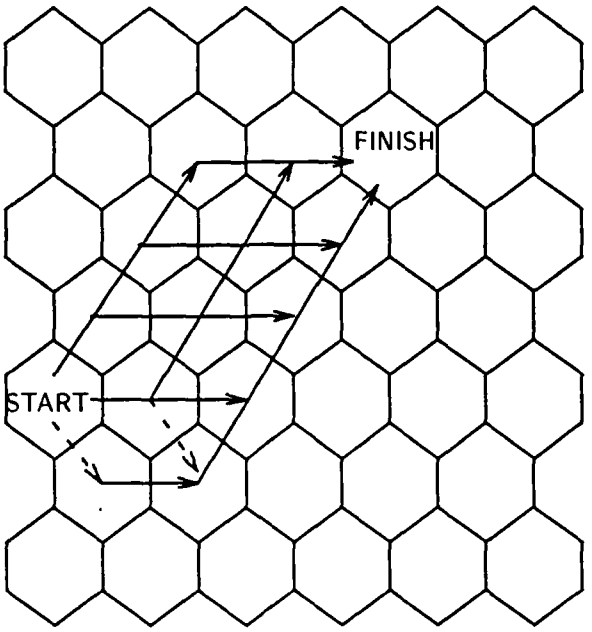

Figure 14. Penalized Indirect EW Movement.

Figure 14 shows the path below an optimal axis for movement. The dashed lines would require extra travel and so would receive a penalty. The solid arrows in Figure 15 would acquire a penalty as compared to the direct movement of the dashed line.

After these weightings have been applied, the optimal solution is chosen based on the lowest rating for the possible directions of movement. Movement rates are then calculated. As mentioned, the weights for all of the above factors are set by the game controllers.

5.2.3 Unit Movement Rates. SELECT_ROUTE also assigns the movement times for units. Unit movement rates are based on the amount of time it would take a unit to traverse from its present hex center to the border and from the border to the next subsequent hex border through that hex center. Although actual movement rates are not calculated, the travel time does directly correlate to the movement rate. Travel time is also called the GRID_TIME. Some of the weighting factors include five levels for different unit missions, eight levels for the type of unit, and six levels for trafficability and weather. The GRID_TIME is based on unit type, the weather and trafficability of the present hex

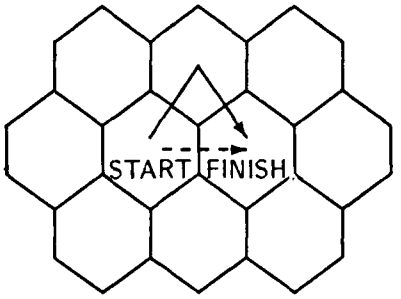

Figure 15. Penalized Indirect EW Movement.

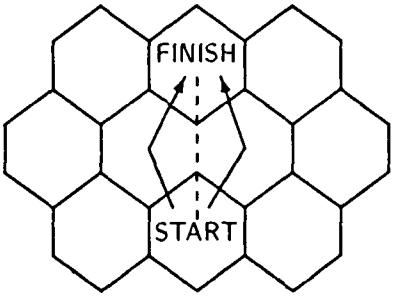

Figure 16. Reduced Direct NS Movement.

and the next hex. Controller supplied weighting factors from the constants file are used so that movement rates can be easily changed by the game controllers. Weather has the least effect on movement. Trafficability is raised to a higher power as compared to weather. Obstacles presently have the most effect on unit movement. Since the values generally range from 0.80 to 1.10 and all of the values are raised to integer powers by the weightings, all of the values are multiplied together for a final GRID_TIME (unit movement time).

The movement time for a unit moving N or S to a hex immediately N or S is reduced by a factor equal to the actual direct movement time, since direct N or S movement is not possible. Figure 16 shows this.

5.2.4 Unit Movement Times. MOVE_IN_GRID simulates unit movement by decrementing the GRID_TIME of a unit to represent the movement of the unit in the hex

(MOVEMENT_TIME). This is calculated based on an average movement rate for units and number of time slices in INITIALIZE. A defender's constant (in hours) MOVE_IN_GRID also adds a defender's number to the GRID_TIME to represent a defending unit's enhancement of his defensive position. A factor of this total GRID_TIME is rolled into the COMBATPOWER of every unit in a defensive posture.

5.2.5 Border Transitions. The general control of land units is based on the amount of time remaining in a hex. When a unit has less than 1.0 time units remaining in a hex or at a border (obstacle), it is considered finished with that hex or obstacle. Control of units and necessary calculations for units in transition from one hex part to another are performed in BORDER_TRANSITION.

The transition from an old hex border to a new hex center involves the determination and selection of the next route and calculation of the movement time. Units are deleted from the old hex and added to the new hex. The unit's new location and GRID_TIME are added as attributes. UPDATE_LOCATION performs the calculation to determine the new location. Units acquiring a WITHDRAW mission are performed separately, since they are being withdrawn from contact and attrition. WITHDRAW requires the checking of adjacent hexes to ensure movement to an adjacent friendly force's hex.

The second possibility for hex transitioning is movement from a hex center to its own border. Only the time to overcome the obstacle is calculated. A unit reaching a boundary is assigned an obstacle reduction time (also called GRID_TIME) based on the attributes of the obstacle at the boundary. Overcoming the obstacle using the unit's own equipment takes considerably more time than if the obstacle is reduced by an engineer support unit. Engineer support must be allocated by the players or units will not receive it. Engineer support occurs in OVERCOME_OBSTACLE.

Once a unit overcomes the obstacle, it enters the hex along that direction of movement, it determines the next possible routes to its final destination, it selects the route of advance, and it acquires the GRID_TIME to move from that border to its center and from the center to the chosen border based on the optimal route selection. This operation may be curtailed if the unit moves into a hex adjacent to an opposing forces hex. Enemy

contact stops unit movement. This occurs whenever a hex is entered that is adjacent to an enemy occupied hex.

5.2.6 Engineer Support. Engineer support is an important component of unit maneuver, particularly if obstacles are present. Engineer support operations are found in the OBSTACLE_OPS package. Engineer units are considered to be accompanying the units which they support and provide the capability to quickly overcome rivers, manmade obstacles and minefields. They may only provide support if their MSN_EFF_DAY is greater than the DAY_NUMBER and the supported unit is actually at a border. Engineer support capabilities directly correlate to the percentage of engineer support that the units receive as specified by the user, and to the controller supplied ENG_SPT_RATE weighting factor from the constants file.

Once an engineer unit has accomplished its mission, the obstacle remains fixed, i.e., broken bridges remain fixed or in place, mine fields are removed, etc. Thus, the trafficability attributes are changed to VG. Units overcoming obstacles with their own capabilities make no changes to the characteristics of the obstacles. Thus, every follow-on unit would also be constrained by the same obstacle.

5.3 Combat

COMBAT makes calls to necessary land combat operations. They include SET_UP which controls ASSESS_CP, SET_ATK, ASSESS_CONTACT and air force operations under APPLY_AFS which includes APPLY_CAS and APPLY_BAI. ATTRITION entails calls to APPLY_FS, APPLY_CP, ATTRIT, DESTROY, and WITHDRAW_UNIT.

5.3.1 Assessing Combat Power. ASSESS_CP calculates the COMBATPOWER of each unit based on the trafficability of the hex, the firepower of the unit and the amount of time the unit has been in the hex if the unit is in a defensive posture. If a unit is not in its final destination defending, then the combat power is merely equal to the unit's inherent firepower. This is discussed further under LB Attrition. The trafficability weighted values are read in from the OLD_CONS file.

5.3.2 Setup for Combat. SET_ATK initializes a unit's IN_CONTACT and IN_ATTRITION attributes to FALSE and sets the unit's hex to ATK if the unit is attacking. This is used in ASSESS_CONTACT and during unit movement.

5.3.3 Assessing Contact. ASSESS_CONTACT initializes the IN_CONTACT and IN_ATTRITION attributes for every hex to false. It determines which hexes contain opposing forces in adjacent hexes and sets the border attributes of both hexes to FEBA = TRUE. This is only done for three sides of the hex, since the operation is done to every hex. When one of two opposing units have an attack mission, then IN_ATTRITION will be true for the two units. Attributes for the units are then set based on the hex in which they reside. When "hexes" are IN_ATTRITION, then attrition will occur between those hexes and units.

5.3.4 Close Air Support. Air Force close air support consists of surviving CAS sorties from the air portion of TWX and is modeled by APPLY_CAS. CAS combatpower is based on the destructive index, the number of sorties, the allocated amount from the CAS record and a weighting factor from the constants file. Each CAS record contains all of the support for each corps with player specified allocation percentages to be applied with specified corps subordinate units. The CAS combat power is applied with the combat power of the hex of the unit being supported. CAS, like BAI occurs only once per cycle and only if the MSN_EFF_DAY for the support is equal to the DAY_NUMBER. The cycle (day or night) is also used to correspond to the correct application time.

5.3.5 Battle Air Interdiction. APPLY_BAI uses target numbers of the objects being targeted (units or hexes). Unit target numbers range from 1 to 500. Hex target numbers range from 10001 to 16540. A hex target number is equal to 10000 + the hex number. APPLY_BAI decrements the FIREPOWER and logistics assets of units being attrited by the BAI air force support. The amount of FIREPOWER decremented from the unit is based on the number of sorties, the destructive index, and a weighting factor from the constants file. If a logistics unit or supply depot has been targeted, then AMMO, POL,

and HARDWARE is destroyed using the same variables and calculations. A delay is added to units that are moving.

Hexes being targeted are checked for bridges. If any bridges exist, they are destroyed and the border trafficability is changed to very poor (VP).

5.3.6 Fire Support. Fire support includes aerial, aviation, field artillery and air defense artillery. All types of support are considered to have been requested when the support is needed by a unit. A unit being blocked by an obstacle would immediately request engineer support, just as a unit in contact would request air force or field artillery fire support. Support will only be provided to units if it is available (sufficient ammunition and firepower) and only if the units requesting the support have been specified by the players to receive the support. Fire support allocation will be partially determined as a result of player interaction.

APPLY_FS adds field artillery, air defense, and aviation support to the COMBAT-POWER of hexes that contain units being supported by the fire support units. The FS_FACTOR provides a weighting factor for the fire support. Support will be provided only against units in adjacent hexes. The amount of SPT_CP (support combat power) applied to the hexes as CP_IN (combat power into a hex from the units) is based on the SPT_PERCENT specified by the player. This support is provided every time step. This only occurs when the units are in contact and attrition is occurring.

5.3.7 Applying Combat Power. APPLY_CP distributes all of the COMBATPOWER of every unit (only if it has ammunition) in a hex equally against all opposing hexes IN_ATTRITION with that hex. This is performed for every hex. The end result is a total COMBATPOWER into (CP_IN) a hex from opposing adjacent hexes and a total COMBBATPOWER out (CP_OUT) from the hex. These combat power factors are used in calculating the attrition of the units.

5.3.8 LB Attrition.

5.3.8.1 General Assumptions. Although many units of the same side may occupy the same hex, opposing units of two sides may never occupy the same hex. Opposing units will be considered in contact whenever any moving unit moves into any hex adjacent to an opponent. This occurs immediately after a unit has cleared the old hex boundary moving into the new hex. Once a unit has crossed the boundary and selected its route of movement, it is committed to that route. This implies that any attacking units will attack any moving unit in its path. These two assumptions are based on the size of the units involved. Although a single point is used to represent the center of mass of a division, it is understood that peripheral reconnaissance and combat units are well ahead of a division's center of mass. It is these lower level, not directly represented units that would be contacting and attriting the opposing forces. Defending units in very well prepared defensive positions would probably not leave those positions unless directed to do so. Red forces probably bypass strongpoints if possible and this is represented so long as two Blue forces are sufficiently far apart so that the Red force never moves to an adjacent hex of a Blue force.

5.3.8.2 Attrition Calculations. The attrition calculations occur in ATTRIT and are performed on every unit in contact with an attacking opposing force. LB attrition is loosely based on the relative combat powers of opposing forces and is based on Lanchester equations. The combat power (CP) of a unit is based on three major factors: its FIREPOWER (FP), the average trafficability of the hex (HT), a FIREPOWER_FACTOR, and the time in hex (TIH). Time in hex applies to defensive units and the FIREPOWER_FACTOR is controller supplied and comes from the constants file. The formula to represent this is(24):

$$CP = FP + (FP \times TIH \times FIREPOWER\_FACTOR \times HT)$$

Hex_trafficability factors presently in use and acquired from the constants file are:

- Excellent (EXC) = 1.00
- Very Good (VG) = 1.02
- Good (GOOD) = 1.04

- Fair (FAIR) = 1.06
- Poor (POOR) = 1.08
- Very Poor (VP) = 1.1

Every hex is surveyed along all sides for adjacent hexes that have opposing forces in attrition. Every hex then distributes the total firepower in the hex to all opposing force's hexes. Distribution of combat power was initially apportioned equally among all opposing forces, but this was later changed to proportional distribution to more correctly reflect reality. These numbers form the basis of the attrition that the units in the hexes will suffer. The formula for attrition (ATR), also called unit loss is(24):

$$ATR(blue) = \frac{CP(red)}{CP(blue)} \times .01 \times CRA$$

where CRA is "Combat ratio adjustment" of the unit based on the following (for blue):

1.0 when CR(blue) is < 1.0.

CR(blue) when 1.0 <= CR(blue) <= 9.0.

9.0 when CR(blue) > 9.0.

where

$$CR(blue) = \frac{CP(red)}{CP(blue)}$$

The same is performed for Red forces. One unit often attrits the other unit faster than it is being attrited particularly if there is a great disparity in the combat powers. These attrition calculations have been successfully used in other training simulations.

Reductions in a unit's AMMO (ammunition) status is also performed based on a unit's ATTRITION value and an AMMO_USE_RATE from the constants file.

5.3.8.3 Results of Attrition. DESTROY performs necessary operations to destroy a unit. When a unit is attrited to FP_DESTROY_VALUE set by the game controller, it is effectively destroyed, its status is provided to the players, and it is no longer playable in the simulation. A player must provide reinforcements or reserves to a unit prior to

its complete destruction. Destroyed units are added to linked lists and output to files BLU_DEST or RED_DEST by the procedure WR_DESTROYED.

5.3.8.4 Unit Withdrawal. A unit's mission may change depending on the attrition it is suffering. WITHDRAW_UNIT ensures that a unit goes to a defensive posture after withdrawing one hex when it is in contact and it is being attrited greater than DELAY_ATTRITION_VALUE for one time period or it has reached its breakpoint. Breakpoints are specified by the players. When a unit withdraws or is forced back, its attrition lines are broken with the opposing forces. It is moved one hex to the rear from its previous direction of movement. The opposing forces then continue on with their previous missions, probably back into contact and attriting again, in effect pursuing the withdrawing unit.

5.4 Logistics

LOG_SPT encompasses calls to DEPOT_LOG and UNIT_LOG_SUPPLY. Although the effects of not having logistics support is probably not felt by a unit until after the fourth day of combat, it is modeled in the the LB. Ammunition, petroleum products and general replacements are the only types of logistics modeled. Depots for each category are established throughout the battlefield at three possible levels: the source at the coast or particular airfield, in the theater area (army group) and in the corps areas. All units have links to their logistics depots and receive support from those depots when stocks get low. Units moving logistics from source to destination are modeled and can be targeted, just as logistics depots can be targeted. BAI is probably the most effective means to target logistics trains or depots. Depots or supply trains that have been hit suffer a reduction in total supplies. Units maneuvering or in combat use logistics at a faster rate. Players may control the movement of the logistics trains from source to final depot and the links between depots and supported units.

In DEPOT_LOG, support units (SPT) moving from the rear provide POL, AMMO, and HARDWARE to its depot once it reaches its final mission destination and if there is a DEPOT located at that location.

In UNIT_LOG_SUPPLY every unit checks for its load of supplies at its assigned DEPOT. If supplies are available, then unit takes its fair share. Fair share is based on a player specified amount (TOTAL_RESUPPLY_PERCENT) for every unit. POL, AMMO and HARDWARE have weighted factors applied to them from the constants file for purposes of varying their importance. TOTAL_AMMO is added to the unit's TOTAL_AMMO, TOTAL_POL is added to a unit's TOTAL_POL, and TOTAL_HARDWARE is added to a unit's FIREPOWER. Once all the supplies have been given out, the DEPOT supplies are zeroed out.

Unit POL reduction occurs in UNIT_OPS under MOVE_IN_GRID. AMMO and FIREPOWER reduction occurs in COMBAT_OPS under ATTRIT as a result of unit attrition.

5.5 Intelligence

INTELLIGENCE controls the calculation of INTEL indexes. It includes the loss of intelligence through time (REDUCE_INTEL), intelligence gathering efforts in front line units at division level, the corps military intelligence brigade (ARMY_INTEL) efforts, and directed special operations forces intelligence gathering efforts (SF_INTEL). The INTEL_INDEX of units and hexes are used to determine the amount of intelligence that is provided to opposing forces players.

The intelligence index of a unit is used to produce an intelligence filter. This determines the amount of intelligence that is provided to opposing forces players. The intelligence index of a unit can range from 0.0 to 1.0, where 1.0 is considered perfect information. The intelligence filter for that unit would range from from 0.0 to 2.0. For example, if the intelligence index was 0.8, the lower range of 0.8 and and upper range of 1.2 (the distance from 1.0 to 0.8 added to 1.0) would be provided to a uniform random number generator. The resulting random number draw (ranging from 0.8 to 1.2) is the intelligence filter and provides greater randomness on the reported numbers as the filter gets further from 1.0. This filter would have a smaller variance as the intelligence index approached 1.0.

If a unit had a firepower of 100, an intelligence index of 0.8, and a resultant intelligence filter of 0.93, then 93 would be reported for the firepower. If further intelligence was gathered against this unit, greater than the amount of lost intelligence, a new index might be 0.88, thus constraining the next intelligence filter to be between 0.88 and 1.12. If no additional intelligence gathering is performed, no additional or changed information is provided to the player.

Additional information is also provided to the player based on three ranges on both sides of 1.0. For ranges from 0.0 to 0.4, the unit is only a suspected unit, and it is shown as infantry. For ranges from 0.4 to 0.8, the unit designator is correct, but it is shown as an armor unit. From 0.8 to 1.0, all unit information is correct. The same calculations apply from 2.0 to 1.6 and on down to 1.0. The opposing forces player has no idea where the intelligence indexes for the enemy unit's stand. Intelligence support only occurs if the supporting unit's mission effective day is greater than or equal to the day number.

Intelligence indexes are also provided for the hexes. As a unit in a hex has its INTEL_INDEX modified, so does the hex. This allows the player to acquire intelligence about hex trafficability and terrain obstacles. The INTEL_INDEX is used directly to report this information.

5.5.1 Intelligence Loss. REDUCE_INTEL performs the reduction of intelligence over time. It is calculated each time slice based on the movement of a unit's INTEL_INDEX away from perfect intelligence (1.0). INTEL_REDUX_RATE from the constants file provides the reduction rate per time slice.

5.5.2 Army Intelligence. ARMY_INTEL models intelligence acquisition in two ways. Units in contact acquire data about the forces they are in attrition with. This simulates the division and corps' subordinate unit intelligence gathering assets. The units having intelligence performed against them have their INTEL_INDEX moved toward 1.0 by the controller specified AVG_INTEL_RATE from the constants file.

The second method for intelligence acquisition is through the corps' military intelligence (MI) brigade, which acquires data based on the player's assignment of support. Units

supported by the MI brigade receive additional intelligence based on being in attrition and the amount of intelligence (INTEL_AMOUNT) the player specifies the unit is to receive. The INTEL_INDEX is modified as discussed above based on the BDE_INTEL_RATE.

5.5.3 Special Operations Forces. SF_INTEL increases the INTEL_INDEX of units in hexes targeted by special operations force (SOF). Players determine the target hexes for the SOF. Any opposing force units located in the targeted hexes will have their INTEL_INDEX modified as discussed above based on the SF_INTEL_RATE.

5.5.4 Air Force Reconnaissance. AF_INTEL increases the INTEL_INDEX of targets against which they fly. It is applied once per cycle against regular combat and logistics units and only if the MSN_EFF_DAY is at least equal to the DAY_NUMBER. AF_INTEL changes the INTEL_INDEX of the TARGET_NUMBER of the unit against which it is flying (if it is a unit) by an amount based on the number of sorties (NO_SORTIES) and the AF_INTEL_RATE.

5.6 Air Interface

AIR_INTERFACE performs some of the required calculations necessary for proper integration of TWX Air with the LB. These include surface-to-air (SAI), interdiction and weather operations.

Every unit type in the LB has an assigned SAI_INDEX. This SAI_INDEX is assigned to every unit and is summed into the hexes which the units occupy. This and the following operations occur prior to completion of the LB program. The SAI_INDEX is used by TWX Air to determine the number of sorties that will survive TWX Air to be passed to the LB.

At the completion of the LB program, hexes and units are classified as IS_INTERDICTION or not, based on the the distance from the enemy FLOT. Although BAI and interdiction targets (both unit and terrain) are treated the same in the LB, TWX Air performs calculations to determine if a target is BAI or interdiction. TWX Air player input is constrained by this check.

For the first time, LB hex weather attributes are based on identical data from TWX Air. A hexes' weather attribute is set according to a weather zone (WEATHER_ZONE) attribute acquired from TWX Air. The hex weather attributes are loaded every day and night cycle prior to the allocation of BAI, CAS, and RECCE.

5.7 Conclusion

This chapter provided discussion of specific details of the detailed design and implementation of the LB program. Of primary importance were the major combat processes. Chapter VI provides a summary, recommendations for further study, and a conclusion.

VI. Conclusion

6.1 Summary

This thesis developed a new land combat model for the Theater War Exercise (TWX). Included in this development was a determination of the proper levels of player participation and the required level of aggregation for the land units. A six step process was used along with application prototyping for program development. The major steps encompassed:

1. Examination of necessary assumptions and constraints. The model had to be PC based and interface with the present air portion of the theater war exercise.
2. Examination of several general purpose and simulation programming languages and data management techniques.
3. Examination of current army doctrine to include large forces operations, levels of warfare, close, deep, rear operations and joint Army and Air Force operations. Necessary unit information, capabilities and operations planning attributes were taken from the latest theory on land combat modeling, simulation of systems theory and US Army Field Manuals and Operating Procedures.
4. Examination of current land combat modeling to include general modeling principles, battle environments, combat processes and unit representations. Of particular concern were the concepts of terrain representation, unit aggregation and necessary combat processes for doctrine encapsulation.
5. Development of requirements specifications based on first four steps of the information gathering process. An object oriented design methodology was used for development of a modifiable, understandable and reliable system. Essential battlefield objects and operations were identified from doctrine and encapsulated in software specifications written in Ada.
6. Code development in Ada. Sound software engineering principles were used to ensure complete, uniform and modular code. Key aspects of the land combat modeling encapsulated in code are:

- Player planning and participation based on the planning and operations faced by an Army Group commander or staff involved in conflict in any area of the world.
- Aggregation of units with focus on division and corps non-divisional units.
- Unit attributes including ten combat and combat support unit types, fire power, combat power, break point, mission, logistics statuses and intel indices. Five mission types can be assigned to units.
- Hexagon based terrain attributes including six levels of terrain trafficability, obstacle and weather characteristics applicable to the six sectors and borders of each hexagon.
- Logistics play to include player allocation and the use and resupply of POL, ammunition and hardware support for each unit.
- Intelligence play based on tactical air reconnaissance from the air portion of the exercise, special operations forces and tactical intelligence gathering operations. Intelligence is provided to players based on player directed operations.
- Land combat support operations to include user assignment and allocation of limited field artillery, aviation, air defense artillery, and engineer support.
- Engineer operations to include automatic mine, bridge and obstacle installation and removal.
- Attrition calculations based on relative combat powers of units. Combat powers are based on unit firepower scores and combat fire support provided by applicable combat support units.
- Unit maneuver based on player inputs for unit missions, mission locations, terrain trafficability, weather, unit characteristics and combat posture.
- Air force operations to include BAI and CAS operations against units and target in hexes.
- Report generation of all key unit interactions and attributes.

The final land battle program has the following characteristics:

- Developed and programmed on a 386 compatible PC in Ada.
- Validated on a PC and Sun workstation, but operational on any system that can compile Ada.
- A time step simulation using hexagonal based terrain and firepower scores.
- Data I/O using flat data files or Oracle data base management system (only on the Sun work station).
- Simulates the doctrinal planning and decision making operations conducted at Army Group level.
- Provides credible land combat processes, unit movement and attrition based on player inputs, unit interactions and terrain characteristics.
- Allows easy insertion of new functions and procedures directly into the main program. By varying the number of time slices per day, any level of detail can be obtained.

The program consists of approximately 4300 lines of highly cohesive, loosely coupled modules of reusable Ada code.

6.2 Recommendations

This thesis provides a stable platform for further development. Although for any given process, detailed research could possibly provide better algorithms, the present results are quite credible. A few particular areas of focus might include:

- Improvements in the air versus air defense artillery interaction. The present configuration uses surface-to-air indexes created by the LB. The indexes are provided to TWX Air, but the creation of the indexes for the units and hexes is only general in nature. This would probably be the largest area for improvement because of its importance.
- Additional discrete combat processes could be added to the program if the time slice were equal to an hour and the players were allowed to plan missions based on days, cycles and hours. This is particularly important for the modeling of nuclear

munitions or detailed time-on-target operations. Several units in a defensive posture could all be given an attack or move mission to start at the exact same time.

- Nuclear operations (artillery, LANCE, Air Force) could easily be modeled, given accurate hourly planning. Modeling could be similar to the present modeling of BAI missions.
- Special Operations Forces operations could be enhanced to include destruction and attrition of enemy forces and obstacles.
- A chemical package could be added to allow use of chemical weapons directly or indirectly against units or terrain. Chemical attributes could be added to hexes (similar to weather) to allow degradation of movement and unit forces as they pass through the terrain. Chemical persistence could be based on the number of hours or time slices.
- Airborne and air assault operations could also be added. This would merely require direct insertion of forces into hexes adjacent to enemy forces.

Significant program runtime improvements could be obtained by moving the entire program to the SUN 386i work station. Much flexibility and a marginal decrease in code would be gained. The Oracle data base management system is fully functional on the SUN and could be used directly from the LB program. Immediate display of unit movement and unit status could be acquired through already written graphics displays provided by other thesis students using Ithaca Software's HOOPS routines (56).

Although results in the land battle certainly seem credible, it could be quite interesting to have an operations research student examine the entire model. Perhaps verification of the algorithms is possible. Fine tuning the constants is also needed. Several hundred simulation runs has pretty much validated the existing code.

A final recommendation is that a thorough analysis of the air portion of TWX be performed. There are many areas in tha' program that are not completely understood, that could be improved upon, and that could be better integrated with the LB program to truly improve the performance of the two. A complete integration of the air and land portions could prove to be a truly powerful learning or analysis tool.

6.3 Conclusion

In conclusion, the LB program is a very powerful tool that is flexible, powerful, easy to understand, and credible. It provides a solid land combat model foundation that contains all of the key processes found in any other model of its size making it capable of simulating the doctrinal planning and decision making that might be conducted at Army Group level. Because of the development methodology, enhancements require minimum effort on the part of the developer and can be easily integrated into the program. There is no other model in the world like it.

Appendix A. *Database Management Systems.*

In the area of data handling there are four basic types of data handling requirements: collection, reduction, generation, and analysis. Data reduction is the process by which any unorganized data is organized. Unorganized data usually causes increasing problems as the amount of data increases. Reduction and reorganization of the data should facilitate any required analysis. Analysis is the extraction of meaningful results from the data, based on some criteria(48).

Data is most often used for input to a model, as information to make decisions in the model, or as outputs from the model which will be analyzed(60). For our purposes here, data collection is primarily the insertion of data into the computer system for use by the simulation, rather than the accumulation of the actual data itself. Although it is usually not the case for war games, as in the case of a high resolution analytic simulation, the data accumulation and input may take several months to a year to accomplish(32). The data to be used by the simulation is typically stored in flat files. Although this decreases the input and output time required to access the data, this type of design emphasizes physical implementation over logical data organization and is very hardware and operating system dependent(13). The requirement to duplicate data items among several different applications or scattered data files may easily cause data integrity and inconsistency problems. Duplication of data might then actually cause an overall reduction in system performance since more overhead may be associated with maintaining the database in a consistent state by ensuring all duplicated data have consistent values. Additionally, the applications programs would have to know the physical location of the data to access it. Should the data structures within these files require any modification, the respective applications programs would also require modifications. This could significantly increase overall maintenance costs for those who must maintain the data and programs. If a new architecture or computer were to be used, the entire simulation might have to be modified, especially for any hardware dependent file management system calls(13).

A database management system (DBMS) allows the simulator to be free of most of the problems associated with data organization at the computer level. Changes in the

storage structure and access strategy of the applications are minimized. Data collection can be aided by the additional features of some DBMSs which include Fourth Generation Language (4GL) application development tools, automated forms management systems and report writers. These tools can be used to design user interface screens and to assist the user to quickly develop functional applications around a menu based system. These tools often incorporate help facilities, on-line editing, and real-time data validation. Properly designed user interfaces can be of great benefit to the user who must input and output the data(13).

Sorting, selection and manipulation of data are some of the requirements that a simulation may require(54). These capabilities are provided by a DBMS. A DBMS not only minimizes many of the problems mentioned earlier, but also provides the following features: data additions, deletions, retrieval, organization, storage and manipulation while providing data security, integrity checking, and minimizing data redundancy and integrity problems. In the area of combat modeling, DBMSs are typically used for manipulating the input and output files.

It is also quite feasible to have direct interaction between the DBMS and the simulation during a simulation run. External data can be used for making decisions during the simulation. Observed values from the run could likewise be stored back to the data base during a run. Variables describing system performance could also be compiled during a run(60). Specific aspects of each simulation project could be stored in the data base for later development, refinement, updating, modification and reuse. Actual system data could be used with the model to determine model parameters and specifications. A DBMS would make this all much easier to do(54).

Database management systems provide great power in reducing the complexities of maintaining data and providing centralized control over data. This makes it much easier to share data among multiple users. A relational model might make the database tables and the data in them much easier to understand and relate to aspects of the actual simulation run. The powers and capabilities of a good DBMS can truly make the simulationist's and analyst's job much easier.

These reasons were sufficient to justify the use of a DBMS and its tools in the development of the new land battle program.

Appendix B. *Land Battle Development with Ada.*

B.1 Introduction.

FORTRAN is probably the most widely used high-level simulation programming language. It along with C are the two languages used in the old version of the Theater War Exercise. Although these languages were sufficient, they lacked many of the capabilities of newer and more powerful languages. This was particularly true considering the size and complexity of the new land battle program. Actual software engineering and development capabilities are quite limited in these languages. For this reason and others, another language was used.

Ada is a general purpose language specified and mandated for use by the Department of Defense in MIL-STD-1815A, dated 17 February 1983 (Ada is a registered trademark of the United States Government, Ada Joint Program Office). Because of the proliferation of hundreds of embedded software languages in the early 1970s, and because of incredible software maintenance costs and problems, Ada was designed as an object-oriented language and software development system that, hopefully, would fully meet the requirements of the Department of Defense. There are many advantages to using Ada for design, development, coding and maintenance. This is discussed further.

B.2 Software Engineering with Ada.

Ada supports all of the basic precepts of good software engineering goals: modifiability, efficiency, reliability and understandability. It meets these goals better than any other languages by inherent application of the principles of software engineering. Those principles include abstraction, information hiding, modularity, localization, uniformity, completeness, and conformability(11, 43, 69). Because Ada is one of the most recently developed languages and its development was formulated with inputs from all of the best software developers world-wide, these goals and principles were built into the language as it was developed(43). Ada helps to enforce the software engineering principles it was designed to support, it is easily maintained, and it is virtually self-documenting(42). Finally, Ada helped introduce the concept of object oriented programming, which provides

many additional advantages over the top-down and data structure design methodologies, although they can still be used in conjunction with object oriented design in software development. A study by Hoover on Ada's use in project development for the United States Marine Corps showed that the use of Ada requires no changes in basic concepts in the Analysis phase of a software development project(11, 39). Although Ada is not required during the Analysis phase, it certainly can be used as a starting point for the formalization of the specifications. The Design phase is more language dependent and it is there where principles and goals of software engineering must be applied. During the Design phase, Ada code can be used as documenting material directly from the specifications. Code may also be used in specifications during validation of the Design phase or through prototyping. Body parts can thus be created much easier in later phases(53). The Coding phase also has no major new changes or requirements(39). During the Coding phase, when the syntax and semantics are verified, the code can almost be used directly from the Design phase. Experience has shown that when Ada interfaces and modules have been previously tested, the resulting assembly of the final software product goes extremely fast as compared to normal languages(51). The final Maintenance phase is also much easier to handle than with most languages. The maintenance documentation is directly related to the documentation code, all of which have an Ada base. Many projects have shown that there were no significant difficulties with learning the important aspects of Ada(53).

B.3 Ada Language Features.

Ada provides many features that are directly usable by the software developer and programmer. These features include the usual programming units such as functions and procedures, as well as generic units that are easily modified for repeated use, tasks for concurrent operations and parallel programming, and packages which are "collections of resources"(11:55). Packages are very reusable, which can ultimately reduce development time. This is also true for generic units, which can significantly reduce the costs of building software. Tasks are programming units that allow concurrent actions, interrupts, the controlling resources and the routing of messages for real time systems(69). Libraries are additional reusable Ada constructs that can be ported between software systems. Another

feature of Ada is the very rigid type casting that it requires. This can truly aid reliability. Strong typing provides a very controlled structure on the language and the program. It is particularly helpful in the early stages of development. Fixing bugs in the early stages of the design phase may save as much as five times as much effort as in the requirements phase, up to 50 times as much effort as in the systems test and up to 100-200 times as much effort in the operations phase(11, 62). Studies by IBM on developed software showed that one-half of the errors were made in the design phase and cost 10-100 times as much to fix during the maintenance phase(51). Over 50% of compilation errors have been found during static validation testing of Ada code(53). Other features that help reliability include constraint checking, enumeration types which allow the use of English type words as types, exception handling, and self-initialization of packages(11). Exception handling is another very powerful feature that is seldom found in other languages. It gives much better run-time capabilities and recovery from improper states. An additional language benefit is the capability for writing the specification only and ensuring that interfaces are sufficient without having to develop the entire system. This is particularly good for the design phase of a software engineering project. Program bodies can also be written separately and like specifications can be compiled separately(69). Compilation of program units finds many programming flaws that can not possibly be found so easily in other languages. Interfacing through specification declarations allow different parts of a software project to be assigned to different programmers. This powerful modularization is a versatile feature. Modular programming can help ensure that program units are highly cohesive and loosely coupled. This allows easy changes and is very good for maintenance of the developing and finished project(62). From the above information, it can be seen that Ada provides many powerful features. In fact, Ada provides all of the capabilities and functionality of any other software language. C, Fortran and Pascal each have only about two-thirds of the overall features of Ada(69).

Not only does Ada have advantages throughout the entire development of projects, the code execution is also fast. Ada code is nearly as fast as Fortran in sequential code and faster in parallel code. This should get even better as compilers become more mature(51).

B.4 Ada as a Simulation Language.

Although Ada was designed primarily for embedded systems and it has no simulation tools, it has been successfully used as a simulation language(12). Object-oriented design using object-oriented languages has produced simulation systems which tend to be much more comprehensible and analyzable than those developed in conventional simulation languages(15). Any simulation language which can be coded in FORTRAN-like or Pascal-like languages could also be implemented in Ada. A given simulation problem could be solved using any one of the basic approaches: event, process or continuous view. It has been shown that it is possible to produce equivalent simulation models in each of the simulation styles. Two process views of simulation are offered in (62).

> Ada is suited for this kind of application, since it supports concurrent processing in accordance with a well defined model. With a wide range of simulation applications, we can get away with only defining a limited number of packages that allow us to express such concepts or data structures as: Simulation time, Events, Event Maintenance, Queues, Queue Ordering Disciplines. The features that are built into Ada such as generics, tasking, and packaging as well as the software engineering concepts, directly supported by the language, make the development of a simulation at least as easy as with specialized simulation languages. In addition, the availability of Ada oriented support environments and design methodologies for the use of Ada, actually extend the modeler's capabilities to develop and maintain simulations (69:145–146).

Discrete process orientation and event orientation simulations have been successfully developed using Ada. Ada thus far appears to be quite effective for implementing simulation programs. There seems to be few limitations encountered by most users (59). Because Ada has real-time concurrency capabilities, it is possible to directly write a simulation model with concurrent processes. Tasking in Ada allows parallel execution of code at the source level(22). This capability is found in very few simulation languages and offers great potential for simulation models. As simulations become larger and more complex, the superior capabilities of Ada will become even more pronounced (14).

Ada brings many advantages to the software engineer that makes its use well worth the effort for the serious programmer(51). There are claims that Ada is perhaps the most

functional of all of the Department of Defense allowed languages and that it definitely minimizes a lot of risk and cost in large projects(51). Ada compilers are slowly becoming available for virtually all types of computer systems. Ada's multiprocessing capabilities, reusable packages, portability, strong type casting and the Department of Defense's requirement that it be used for all new software systems will contribute significantly to its use as a simulation language. For all of these reasons, Ada was chosen as the design and development language for the LB program.

Appendix C. *Objects and their Attributes.*

This appendix discusses the objects and their attributes used in the LB program. At the highest level of abstraction are the following objects.

- DATA: This includes all data encapsulated in the databases maintained for the Theater War Exercise. This data may be the result of direct input by the users, program manipulation or program creation. Data must be input into the LB and out of the LB for database update and report creation. All of the objects can be considered data of one type or another and will be processed by a DATA_IO package or subprogram. Data includes the attributes of the objects described in subsequent steps. Reports are provided by the 4th Generation Language tools of of the database management system or the REPORT_WRITER procedure. They are required periodically and at the end of cycles. Filtered intelligence of enemy units and activities is provided to the players by WRITE_DATA based on a unit's INTEL_INDEX.

At the next lower layer of abstraction are the actual *objects, structures and types* manipulated by the higher levels. These objects are:

- UNITS: Units are the basic operational objects in the land battle. They move and attrit other units. They request support and they may provide support, depending upon their type. The two general classification of units are combat and logistics. Required attributes include:

  1. TARGET_NUMBER: This is a key number for every unit.
  2. CORPS_ID: The corps to which this unit belongs.
  3. UNIT_DESIGNATOR: This is the actual unit name designator for each unit.
  4. TYPE_OF_UNIT: This indicates the types of units, e.g. Armor, Infantry, Cavalry, Field Artillery, Air Defense Artillery, Engineer, and Air Force. Initially only these units will be represented by the simulation.
  5. FORCE: Whether a unit is a RED, BLUE or NEUTRAL force. Used to determine opposing forces.

6. MSN_EFF_DAY: The effective day on which this unit will actively move or provide support. This has no effect on being in attrition with an enemy unit.
7. MISSION: This specifies the types of missions a unit may be required to perform, e.g. Attack, Defend, Withdraw, or Support.
8. PRESENT_LOCATION: Specifies the current location of a unit. This is used in conjunction with the hex terrain system for tracking unit movement.
9. MISSION_LOCATION: This is the location of the unit's mission. This may also initiate unit maneuver if MISSION_LOCATION is different from PRESENT_LOCATION.
10. REGION: Location of a unit in a hex CENTER or at the BORDER.
11. HEX_DIR: The six directions of movement for a unit. Specifically states if a direction is allowed for unit movement.
12. MOVE_ALLOWED: A boolean value for each of the six sides of the hex a unit is presently occupying indicating whether the unit is allowed to move in that direction.
13. FIREPOWER: This is a firepower score. It is percentage of full combat capability or an index used for comparison with other unit's strengths based on an aggregation of the unit's inherent combat capability.
14. COMBATPOWER: Total combatpower of a unit. Includes firepower, defensive posture, and terrain characteristics. It is always at least equal to a unit's FIREPOWER.
15. ATTRITION: The amount of attrition that a unit suffered during one time period.
16. IN_CONTACT: A boolean of whether the unit is in contact with opposing forces.
17. IN_ATTRITION: A boolean of whether the unit is attriting with opposing forces. One unit must be attacking for IN_ATTRITION to be true.
18. TOTAL_LOG: This is only one of the logistics status codes of a unit.

19. LOG_RESUPPLY_PERCENT: This is one of three amounts of support that a unit is to receive from his logistics support unit. The three types of logistics are AMMO, POL and HARDWARE.

20. INTEL_INDEX: This indicates the amount of intelligence that enemy units have acquired about this unit.

21. BREAKPT: This is a percentage of combat power which specifies the breakpoint of a unit. When a unit is attrited to this point, it will withdraw from contact.

22. GRID_TIME: The amount of time a unit requires to traverse a portion of a hex or the amount of time a defending unit has been defending in hex.

23. SPTED_UNITS: These are the units (an array called UNITS_TO_SPT) this unit provides support to. This applies specifically to artillery, air defense, engineer and air force units, which would support other units. SPT_PERCENT shows the amount of support each SPTED_UNIT is to receive.

- BAI_TYPE, CAS_TYPE, RECCE_TYPE. These items contain necessary TARGET_NUMBER or CORPS_ID, MSN_EFF_DAY, NO_SORTIES, AIRCRAFT, DESTRUCTIVE_INDEX or UNIT_DESIGNATOR, and arrays of supported units information for air force assets to affect ground forces.

- SF_INTEL_TYPE: This is an array of records which contain the FORCE, MISSION_LOCATION and SPT_AMOUNT attributes for application of special operations forces against the enemy.

- GRID_TYPE: This object contains the data specifications for the actual terrain representation and features that units may act through and upon. Each grid represents a unit of land. Units will not only be affected by characteristics of actual grids, but must also be represented in the grids. Land unit locations can therefore be easily controlled and tested. Minimum attributes include:

  1. GRID_LOCATION: This is a grid location based on a grid system. This uniquely specifies every grid square in the Land Battle.

2. TARGET_NUBMER: This is a unique identifier for each hex. It is equal to 10000 + the hex number.

3. WEATHER: This specifies the weather attribute for each hex: EXC, VG, GOOD, FAIR, POOR and VP to represent clear, foggy, raining, night, storming or snowing.

4. FORCE: The force type occupying the hex, either BLUE, RED or NEUTRAL.

5. IN_CONTACT: This is true if a hex has units in contact with an adjacent hex having opposing forces.

6. IN_ATTRITION: If IN_CONTACT is true and any of the units has an attack mission.

7. CP_OUT: Total combat power of all units in a hex which can be applied to opposing hexes.

8. CP_IN: Total combat power from all adjacent hexes directed into a hex.

9. ATTRITION: The amount of attrition that all the units of a hex suffered.

10. NEXT_UNIT: This points to the list of all UNITS presently in a grid.

11. SIDE_DEF: The definitions of the six borders of the hex. Each border includes the following:

    (a) OBSTACLES_ITEM: This specifies objects in a grid square that may inhibit/expedite unit movement.

    (b) OBSTACLE_DIF: OBSTACLES difficulties are EXC, VG, GOOD, FAIR, POOR and VP. The represent levels of difficulty of man_made structures, minefields, craters and lack of bridges.

    (c) TRAFFICABILITY: This specifies the land maneuverability for every grid square: EXC, VG, GOOD, FAIR, POOR and VP to represent improved roads, cross country, forests, swamps and water. This is required for movement rates for all land units. TRAFFICABILITY extends in a pie shape from the boundary to the center of hex.

    (d) FEBA: This is true if the unit is in contact with enemy forces.

Appendix D. *Operations and Their Attributes.*

This appendix discusses the general operations and their attributes as they were initially designed for the LB program. It is for use as a reference to the initial design process and should not be used for the present implementation of the LB program. Most of the procedures here are called by the main procedure. More thorough, complete and accurate discussion of the final operational program can be found in Chapter V.

The first level of abstraction includes the following operations:

- DATA_INPUT: One of the first operations performed under the LB. It moves data from the database into appropriate data structures of the LB program. All objects have attributes represented by database elements. Required attributes may also be initialized.
- DATA_OUTPUT: One of the last operations performed by the LB program. Current data from the LB is written to the appropriate databases.
- REPORT_WRITER: Writes all of the reports for the player's use. May use data directly from the database or acquire data via the relational database system. This will occur after every simulation run. Necessary reports are sent to the printer. Filtering for intelligence reports is performed based on the INTEL_INDEX for any given unit.

The primary operations of the second level of abstraction are:

- ASSESS_CP. Assesses the full combat power of a unit or a grid. Takes into account the trafficability of the hex, the firepower of the unit, and the amount of time a defending unit has spent in the defensive posture. If a unit is not defending or in his defensive position then the combat power is equal to his firepower.
- ASSESS_CONTACT. This initializes the IN_CONTACT, CP_IN, CP_OUT, and FEBA parameters for units and hexes. It determines which hexes containing units have adjacent hexes that contain opposing forces. This must be true for two ground combat forces to attrit one another. If either of the opposing forces are attacking, then

IN_ATTRITION is true for hex and for the units. This means ATTRITION will take place among the hexes and the units.

- APPLY_FS. This provides fire support and includes aerial, aviation, field artillery and air defense artillery. A portion of these units firepower is added to the CP_OUT of any unit they support. The portion is set by the player and can range from 0 to 100 percent of the unit's total firepower.
- APPLY_AFS. This applies air force support directly to units.
- APPLY_CP. Distributes combat power to all adjacent units in attrition with this unit. All units in a hex have their combined firepower added to the entire hex CP_OUT. This is then applied equally (added to that unit's CP_IN) to all adjacent hexes that are in attrition with this hex (hex that has a unit with an attack mission i.e. IN_ATTRITION = true). Thus every unit also receives a combined CP_IN from all opposing forces in adjacent hexes.
- ATTRITION. The process of applying attrition to units. Attrition is based on the combat ratios and adjustment factors, particularly the CP_IN and CP_OUT ratios. Hex attrition is first calculated and then an appropriate amount is applied to every unit in that hex. This is performed every time step of the simulation. Minimum attrition for a unit in combat is one percent. The unit's firepower is reduced by an appropriate amount based on the attrition.
- DESTROY. Units that are attrited to a firepower of five are no longer effective. They are destroyed by removing them from the simulation. The unit is no longer playable. Results are reported to the players.
- MANEUVER. This is the overall controlling procedure for all unit movement. Included in this are the required checks for movement (units not in contact and not at final destination), BORDER_TRANSITION, MOVE_IN_GRID, and OVERCOME_OBSTACLE.
- DETERMINE_ ROUTE. This determines the possible routes of movement based on initial and final destinations. All six of the directions are checked based on the

general east-west or north-south general axis of advance. Movement is constrained as described in Chapter V.

- SELECT_ROUTE. A selection of the best of several possible routes. Selection is based on weather, trafficability, and obstacles. Values are assigned from 0 to 5 for ranges of each (i.e. EXC to VP). Weather has a weighting of one, trafficability a weighting of two and obstacles a weighting of three times the value assigned to the attribute. The lowest total based on movement from the present hex center, through the border and to the next hex center is used as the selected direction. A penalty is applied to any unit moving considerably outside of the straight line to his final destination to model what a unit probably would not do. Unit movement times are also calculated based on these attributes and the type of unit. A reduction in movement time is factored in for a movement to a hex directly north or south of the present hex and for one movement directly east or west. Further discussion is in Chapter V.
- MOVE_IN_GRID. The control of unit movement in grids and at obstacles. A standard amount of movement time is subtracted from every unit's GRID_TIME to model the unit's movement in a hex or overcoming an obstacle. If a unit is defending at a final mission location, then GRID_TIME is added for additional firepower purposes.
- BORDER_TRANSITION. This controls units as they cross hex borders, either from hex to border or from border to hex. Times to overcome any obstacles are also calculated. DETERMINE_ROUTE and SELECT_ROUTE is performed when border to hex movement occurs. As the unit leaves the old hex, it is deleted from the linked list maintained by that hex and is added to the new hex. A new GRID_TIME is also calculated for the unit.
- UPDATE_LOCATION. Updates a unit's PRESENT_LOCATION attributes as it moves from one hex to the next.
- OVERCOME_OBSTACLE. A unit overcomes obstacles at its own very slow rate unless the players specify engineer support. If engineer support is designated for a unit, the percentage specified by the player is provided to the needing unit. Engineer

support is about six times faster than what a unit can perform. Engineer support is also permanent. Things that are fixed, remain fixed. If no engineer support is provided, then follow on units must overcome the obstacle again.

The third level of abstraction includes operations that manipulate the access types (pointers) of units or hexes and include:

- INITIAL_PTR. This initializes the grid-to-unit pointers. It is performed at the beginning of the simulation run.
- ADD. This adds a unit to the linked list of units maintained by the hexes as the units move from border to hex.
- DELETE. This deletes a unit from a hexes' linked list as the unit leaves the hex. If a unit is the last unit departing a hex, then the hex becomes NEUTRAL.

Further discussion of these operations can be found in Chapter V.

Appendix E. *Problems with MS-DOS.*

This appendix discusses some of the problems encountered with the MS-DOS operating system. Initial difficulties began with the JANUS Ada 2.1.2 compiler. It was incapable of compiling the program in its entirety. The compiler symbol space was insufficient to hold all the symbols. This necessitated ordering the JANUS Ada 386-to-DOS compiler version 2.1.3. This compiler had four times the symbol space and double all of the other table spaces as compared to Version 2.1.2.

Fortunately, in the interim a VERDIX Ada compiler was available for the SUN 386i. This compiler allowed all packages and their specification to exist in the same file. No changes were made to the existing DOS code. Only the file names had to be changed, since UNIX is case sensitive, and DOS is always capitalized. The VERDIX compiler is very fast and allowed completion of the LB program prior to the arrival of the JANUS 386 compiler.

Initial problems encountered under DOS was the 640k RAM limit and the single 64k data segment. JANUS requires that heap space and stack space not exceed 64k. The heap is where all access objects are allocated. All local variables go in the stack space. Even the allocation of very small hex grids caused the two spaces to meet causing STORAGE_ERROR. Since the original program had all units as access types, the first major conversion was to change the unit's data structure to a BIGARRAY. This was also done with the hex grid system. The BIGARRAY provided by JANUS allows the data to be stored outside of the 64k data segment. The program size (up to maximum memory size under memory model 1), the BIGARRAYs, the constant segment (64k), and the data segment still cannot exceed 640k. This was an original constraint of this thesis, so that it would operate under a Zenith Z-158.

As a result of the above problems, the hex system was set at 65 East-West by 40 North-South and the total number of units was limited to 450.

Bibliography

1. R. J. Abbott. Program design by informal English descriptions. *Communications of the ACM*, 26(11):882–894, November 1983.
2. Osman Balci. The implementation of four conceptual frameworks for simulation modeling in high-level languages. In *1988 Winter Simulation Conference Proceedings*, pages 287–295. The Winter Simulation Conference, San Diego CA, 1988.
3. C.B. Barfoot. The Lanchester attrition-rate coefficient: Some comments on Seth bonder's paper and a suggested alternate method. *Operations Research*, 17:888–894, 1969.
4. John A. Battilega and Judith K. Grange, editors. *The Military Applications of Modeling*. Air Force Institute of Technology Press WPAFB,Ohio, 1978.
5. William E. Biles. "Introduction to Simulation". In *Proceedings of the 1987 Winter Simulation Conference*, pages 761–764. San Diego, CA: The Winter Simulation Conference, 1987.
6. John R. Bode. Indices of effectiveness in general purpose force analysis. In *Developments in Theater Level War Games*, pages 1–65. Military Operations Research Society, December 1974. (ADA039901).
7. Seth Bonder. The Lanchester attrition-rate coefficient. *Operations Research*, 15:221–232, 1967.
8. Seth Bonder. The mean Lanchester attrition rate. *Operations Research*, 18:179–181, 1970.
9. Seth Bonder. An overview of land battle modeling in the US. *Proceedings 13th US Army Operations Research Symposium*, pages 73–87, November 1974.
10. Grady Booch. Object oriented design. *Ada Letters*, 1(3):64–76, March 1982.
11. Grady Booch. *Software Engineering with Ada*. Benjamin and Cummings Publishing Co, Inc, Menlo Park, CA, 1987.
12. Jesus Borrego et al. A space logistics simulation implementation in Ada. In *Proceedings of the 1988 Winter Simulation Conference*, pages 761–764. The Winter Simulation Conference, San Diego CA, 1988.
13. Michael D. Brooks. Developing a database management system and air simulation software for the Theater War Exercise. Master's thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1987. (ADA189681).
14. Giorgio Bruno. Rationale for the introduction of discrete simulation primitives in Ada. In *Simulation in Strongly Typed Languages: Ada, Pascal, Simula...Simulation Series*, volume 13, pages 10–15. Simulation Councils, Inc, La Jolla, CA, 1984.

15. Stephanie Cammarata et al. Dependencies and graphical interfaces in object-oriented simulation languages. In *Proceedings of the 1987 Winter Simulation Conference*, pages 507–517. The Winter Simulation Conference, San Diego CA, 1987.

16. Thomas A. Cardwell. *Command Structure for Theater Warfare, The Quest for Unity of Command.* Maxwell AFB: Air University Press, September 1984.

17. Gordon M. Clark. The combat analysis model (U). *Proceedings 24th Military Operations Research Symposium*, pages 171–180, November 1965.

18. BDM Corporation. The combat II model. In *Developments in Theater Level War Games*, pages 1–46. Military Operations Research Society, December 1974. (ADA039901).

19. B.J. Cox. The message/object programming model: A small change at a deep conceptual level. In *IEEE Proceedings of the Softfair: A Conference on Software Development Tools, Techniques and Alternatives*, pages 51–60, July 1983.

20. James D. Craig. The effect of uncertainty on Lanchester-type equations of combat. Master's thesis, Naval Postgraduate School, Monterey, CA, September 1975. (ADA017550).

21. John Cushman, Wayne Hughs, et al. On representing warfare. Technical report, Office of the Joint Chiefs of Staff, Washington, DC, April 1986.

22. Carolyn K. Davis et al. Automatic development of parallel simulation models in Ada. In *Proceedings of the 1988 Winter Simulation Conference*, pages 339–343. The Winter Simulation Conference, San Diego CA, 1988.

23. Reed E. Davis and Cheryl L. Seyboth. A review of selected models. Technical Report DNA 5967Z, Defense Nuclear Agency, Defense Nuclear Agency, Washington, DC 20305, February 1982.

24. Department of Military Strategy, Planning and Operations. *US Army Group Operations (CONEWAGO 88), Academic Year 1988.* Carlisle Barracks, PA: United States Army War College, 1988.

25. Department of the Air Force. *Combined Air Warfare Course, Theater War Exercise, Users' Handbook 1987.* Combat Employment Institute, Maxwell AFB, AL, 1987.

26. Department of the Army. *Combat Intelligence.* Washington: HQ, Department of the Army, October 1973. FM 30-5.

27. Department of the Army. *Combined Arms Operations.* Fort Leavenworth, KS: Combined Arms and Services Staff School, 1985.

28. Department of the Army. *Organization of the Army in the Field.* Fort Leavenworth, KS: Combined Arms and Services Staff School, 1985.

29. Department of the Army. *Operations.* Washington: HQ, Department of the Army, May 1986. FM 100-5.

30. EVB Software Engineering, Inc. *An Object Oriented Design Handbook for Ada Software.* EVB Software Engineering, Inc., 1985.

31. Henry J. Friedman. Automated development of Lanchester attrition coefficients for the Joint Theater Level Simulation System. *Operations Research Society of America*, April 1988.

32. Major Michael W. Garrambone. Personal comments. August 1989.

33. Geoffrey Gordon. *System Simulation.* Prentice Hall Inc, Englewood Cliffs New Jersey, 1969.

34. James K. Hartman. *Lecture Notes in Aggregated Combat Modelling.* James K. Hartman, 1985.

35. James K. Hartman. *Lecture Notes in High Resolution Combat Modelling.* James K. Hartman, 1985.

36. Charles F. Hawkins. Modeling the breakpoint phenomena. *Signal*, pages 38–41, July 1989.

37. R.L. Helmbold. Historical data and Lanchester's theory of combat. Technical Report (AD480975), Combat Operations Research Group, P.O. Box 116, Fort Belvoir, VA, 22060, July 1961.

38. R.L. Helmbold. Historical data and Lanchester's theory of combat part ii. Technical Report (AD480109), Combat Operations Research Group, P.O. Box 116, Fort Belvoir, VA, 22060, August 1964.

39. Thomas B. Hilburn. The use of Ada in the teaching of data structures. *Ada Software Engineering Education and Symposium*, pages 253–252, June 1988.

40. Francis P. Hoeber. Case studies in military applications of modeling. Technical Report F33600-79-C-0525, Hoeber Corporation, Arlington, VA 22207, May 1980.

41. J. Honig et al. *Review of Selected Army Models.* Models Review Committee: Department of the Army, May 1971. (AD887175).

42. Kim L. Hoover. Integration of Ada software engineering training within the Ada system development process. *Ada Software Engineering Education and Symposium*, pages 164–186, June 1988.

43. Bruce W. Johnston. An Ada based software engineering project course. *Ada Software Engineering Education and Symposium*, pages 83–106, June 1988.

44. Edward P. Kerlin et al. *Computerized QUICK GAME: A Theater-Level Simulation (U) Volume 1 Models.* McLean, VA: Research Analsysis Corporation, November 1967. (AD 387510).

45. Naim A. Kheir. *Systems Modeling and Computer Simulation.* Marcel Dekker Inc, New York, 1988.

46. F. W. Lanchester. *Aircraft in Warfare: The Dawn of the Fourth Arm.* Constable, London, 1916.

47. Lawrence J. Low. *Theater-level and Analysis Workshop for Force Planning.* Operations Research Program, Arlington, VA, September 1977. (ADA101846).

48. Francis F. Martin. *Computer Modeling and Simulation.* John Wiley & Sons, Inc, NY, 1968.

49. James P. LTC McGourin. Comments provided on rough draft of thesis. September 1989.

50. William W. Mendle and LTC Floyd T. Banks. *Campaign Planning (U).* Carlisle, PA: Army War College, Strategic Studies Institute, January 1988. (AD-B121 770L).

51. MILSTAR. Milstar Ada study in software engineering issues, phase II separations. Technical report, September 1984.

52. Paul L. Olsen. A maneuver-oriented model of conventional ground combat. In *Developments in Theater Level War Games*, pages 400–477. Military Operations Research Society, December 1974. (ADA039901).

53. Michel Papaix and others. Two years of Ada experiments: Lessons and results. In *Ada: Managing the Transition.*, pages 67–78. Cambridge: Cambridge University Press, 1986.

54. Alan B. Pritsker. *Introduction to Simulation and SLAM II (Third Edition).* A Halsted Press Book, New York, 1986.

55. J.S. Przemieniecki. *An Introduction to Mathematical Methods of Defense Analyses.* Air Force Institute of Technology, 1986.

56. Darrell Quick. A graphics interface for the Theater War Exercise. Master's thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1988.

57. Mark A. Roth. Personal conversation. November 1988.

58. Arthur L. Schoenstadt and Samuel H. Parry. Toward an axiomatic generalized value system. Technical report, March 1986.

59. Sallie Sheppard et al. Simulation in Ada: An implementation of two world views. In *Simulation in Strongly Typed Languages: Ada, Pascal, Simula ... Simulation Series*, volume 13, pages 3–9. Simulation Councils, Inc, La Jolla CA, 1984.

60. C.R. Standridge and A.A.B. Pritsker. Using data base capabilities in simulation. In *Progress in Modeling and Simulation*, pages 347–365. Academic Press, NY, 1982.

61. James G. Taylor. *Force-On-Force Attrition Modelling.* Military Applications-Section, Operations Research Society of America, 1981.

62. Brian W. Unger et al. *Simulation Software and Ada.* Simulation Councils, Inc, La Jolla CA, 1984.

63. United States Army Material Development and Readiness Command. *Engineering Design Handbook, Army Weapon System Analysis Part 2.* Department of the Army, October 1979. (ADA055007).

64. United States Army Training and Doctrine Command. *Tactical Command Control.* Fort Monroe, VA: HQ, United States Army, June 1980.

65. United States Comptroller General. *Models, Data, and War: A Critique of the Foundation For Defense Analyses.* United States General Accounting Office, March 1980.

66. United States Readiness Command. *General Operating Procedures for Joint Attack of the Second Echelon (J-SAK).* MacDill AFB: United States Readiness Command, December 1984. USREDCOM Pam 525-8, TRADOC Pam 525-45, TACP 50-29.

67. U.S. Army Command and General Staff College. *Coordinating Draft, Large Unit Operations.* Fort Leavenworth, Kansas, September 1987. FM 100-6.

68. Michael H. Vernon. *(U) Air Interdiction: Joint Coordination Issues for the United States Army and Air Force Conducting Coalition Warfare within the NATO Theater of Operations.* Fort Leavenworth, KS: Army Command and General Staff College, School of Advance Military Studies, April 1986. (AD-A174 193).

69. Robert H. Wallace. *Practitioner's Guide to Ada.* Intertext Publications, Inc. McGraw-Hill, Inc, New York, 1986.

70. Kenneth R. Wilcox. Extending the user interface for the Theater War Exercise. Master's thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1988.

Vita

Captain Marlin A. Ness [redacted] He graduated from William Byrd High School in Vinton, Virginia and attended the United States Military Academy at West Point, New York. He graduated in 1980 with a Bachelor of Science degree. He has been assigned as platoon leader with the 2d Armored Division at Fort Hood, Texas, as the Communications Electronics Staff Officer of the V Corps Artillery, and as the Company Commander of A Company, 17th Signal Battalion, 22d Signal Brigade in Frankfurt, West Germany. In September 1988, he was assigned to the Air Force Institute of Technology.

[redacted]

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/ENG/90J-01	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Air Force Wargaming Center 8b. OFFICE SYMBOL (If applicable) AUCADRE/WG	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Maxwell AFB, AL 36112-5532	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
A New Land Battle for the Theater War Exercise (UNCLASSIFIED)

12. PERSONAL AUTHOR(S)
Marlin A. Ness, B.S., Captain, US Army

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Thesis	FROM _____ TO _____	1990 June 1	111

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Wargaming, Land Combat Modeling, Database, Theater Warfare
12	5		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)
Thesis Advisor: Mark A. Roth, Major, USAF
Associate Professor of Electrical Engineering and Computer Science

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Mark A. Roth, Major, USAF	22b. TELEPHONE (Include Area Code) (513) 255-3576 22c. OFFICE SYMBOL AFIT/ENG

DD Form 1473, JUN 86 *Previous editions are obsolete.*

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Block 19: Abstract.

The Theater War Exercise (TWX) is a four day, theater level, two sided, airpower employment decision-making exercise run at the Air Force Wargaming Center, Maxwell AFB, Alabama. Decisions made by the exercise participants are fed into TWX's air and land battle simulation programs, which then simulate the employment of the air power strategy, doctrine, and war fighting principles inherent in those decisions. TWX was limited for use as a potential joint exercise because of the existing land battle simulation. The old land battle was a standalone simulation whose only interface to the air battle was the passing of sortie information through an intermediate database relation. Additionally, a carefully prepared programming script was used to ensure that the land units performed reasonably. This made it impossible for the land game to react to the air play in any way except through a slowing of land units when on their preprogrammed paths.

The goal of this thesis effort was a complete development of a new land combat model program. This was accomplished through a determination of the proper levels of player participation and the required level of aggregation for the land units. A six step object oriented design process was used along with application prototyping for program development.

An aggregated interactive theater-level land combat model and its implementation in Ada is described. The model simulates doctrinal planning and decision making operations conducted at Army Group level. It provides credible land combat processes, unit movement and attrition, logistics, and intelligence based on player inputs, air–land unit interactions and terrain characteristics.

Overall, the program is an easily modified, easily configured simulation that is potentially useful at any level of combat. It is a very powerful tool that is flexible and credible. It provides a solid land combat modelling foundation that contains all of the key processes found in any other model of its size. Because of the object oriented development methodology, enhancements require minimum effort on the part of the developer and can be easily integrated into the program.